DIGITAL
multimedia



Nigel Chapman and
Jenny Chapman

DISH TECHNOLOGIES L.L.C., EXHIBIT 2008, p. 1
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

WILEY

DIGITAL
multimedia

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 2
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

WORLDWIDE
SERIES IN
COMPUTER
SCIENCE



Series Editors **Professor David Barron, Southampton University, UK**
Professor Peter Wegner, Brown University, USA

The Worldwide series in Computer Science has been created to publish textbooks which both address and anticipate the needs of an ever evolving curriculum thereby shaping its future. It is designed for undergraduates majoring in Computer Science and practitioners who need to reskill. Its philosophy derives from the conviction that the discipline of computing needs to produce technically skilled engineers who will inevitably face, and possibly invent, radically new technologies throughout their future careers. New media will be used innovatively to support high quality texts written by leaders in the field.

Books in Series
Ammeraal, *Computer Graphics for Java Programmers*
Ammeraal, *C++ for Programmers 3rd Edition*
Barron, *The World of Scripting Languages*
Ben-Ari, *Ada for Software Engineers*
Gollmann, *Computer Security*
Goodrich & Tamassia, *Data Structures and Algorithms in Java*
Kotonya & Sommerville, *Requirements Engineering: Processes and Techniques*
Lowe & Hall, *Hypermedia & the Web: An Engineering Approach*
Magee & Kramer, *Concurrency: State Models and Java Programs*
Peters, *Software Engineering: An Engineering Approach*
Preiss, *Data Structures and Algorithms with Object-Oriented Design Patterns in C++*
Preiss, *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*
Reiss, *A Practical Introduction to Software Design with C++*
Schneider, *Concurrent and Real-time Systems*
Winder & Roberts, *Developing Java Software 2nd Edition*

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 3
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

DIGITAL MULTIMEDIA

Nigel Chapman and
Jenny Chapman

JOHN WILEY & SONS, LTD
Chichester • New York • Weinheim • Brisbane • Singapore • Toronto

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 4
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Copyright © 2000 by John Wiley & Sons Ltd,
Baffins Lane, Chichester,
West Sussex PO19 1UD, England
National 01243 779777
International (+44) 1243 779777
E-mail (for orders and customer service enquiries):
cs-books@wiley.co.uk
Visit our Home Page on http://www.wiley.co.uk
or http://www.wiley.com

Reprinted November 2000, July 2001, January 2002

Cover illustration: Pablo Picasso's "La Guitare" 1913 © Succession Picasso/DACS 2001. Reproduced with permission from The Museum of Modern Art, New York.

PICASSO, Pablo
Guitar. Céret (after March 31, 1913)
Pasted paper, charcoal, ink, and chalk on blue paper, mounted on ragboard, 26½ x 19½" (66.4 x 49.6 cm).
The Museum of Modern Art, New York, Nelson A. Rockefeller Bequest.
Photograph © 1999 The Museum of Modern Art, New York.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency, 90 Tottenham Court Road, London, UK, W1P 9HE, without the permission in writing of the publisher, with the exception of any material, supplied specifically for the purpose of being entered and executed on a computer system for exclusive use by the purchaser of the publication.

Neither the author nor John Wiley & Sons, Ltd accept any responsibility or liability for loss or damage occasioned to any person or property through using the material, instructions, methods or ideas contained herein, or acting or refraining from acting as a result of such use. The author and publisher expressly disclaim all implied warranties, including merchantability or fitness for any particular purpose.

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Ltd is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Nigel Chapman and Jenny Chapman have asserted their right under the Copyright, Designs and Patents Act 1988 to be identified as the authors of this work.

John Wiley and Sons Ltd., Baffins Lane, Chichester, West Sussex,
UK PO19 1UD.

Other Wiley Editorial Offices

John Wiley & Sons, Inc., 605 Third Avenue,
New York, NY 10158-0012, USA

Weinheim • Brisbane • Singapore • Toronto

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 0-471-98386-1

Produced from authors' own PostScript files
Printed and bound in Great Britain by Biddles Ltd, Guildford and King's Lynn.
This book is printed on acid-free paper responsibly manufactured from sustainable forestry, in which at least two trees are planted for each one used for paper production.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 5
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Contents

	Preface	**xi**
1	**Introduction**	**1**
	Applications of Multimedia	2
	Delivery of Multimedia	6
	Historical Context	7
	Multimedia Production	9
	Terminology	11
	Interactivity	13
	Social and Ethical Considerations	17
	A New Medium	30
	Further Information	31
	Exercises	32
2	**Enabling Technologies**	**34**
	Digital Representations	35
	Hardware Requirements	45
	Software	50
	Networks	53

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 6
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Standards 60
Further Information 63
Exercises 64

3 Introduction to Computer Graphics 66
Vector Graphics and Bitmapped Graphics 68
Combining Vectors and Bitmaps 74
File Formats 80
Further Information 84
Exercises 85

4 Vector Graphics 86
Fundamentals 87
Shapes 91
Transformations and Filters 101
3-D Graphics 104
Further Information 119
Exercises 119

5 Bitmapped Images 121
Resolution 121
Image Compression 125
Image Manipulation 133
Geometrical Transformations 148
Further Information 151
Exercises 151

6 Colour 153
Colour and Science 154
RGB Colour 156
Other Colour Models 166
Channels and Colour Correction 175
Consistent Colour 177
Further Information 183
Exercises 183

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 7
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

7 Characters and Fonts **185**

Character Sets 186

Fonts 196

Further Information 217

Exercises 217

8 Layout **219**

Text in Graphics 220

Markup 222

Text Layout Using HTML and CSS 234

Portable Documents 262

Further Information 266

Exercises 267

9 Hypertext **269**

A Short History of Hypertext and Hypermedia 270

The Nature of Hypertext 272

Links 274

Navigation and Structure in Hypertext 291

Further Information 297

Exercises 298

10 Video **300**

Digitizing Video 302

Video Standards 305

Introduction to Video Compression 312

QuickTime 322

Digital Video Editing and Post-Production 325

Streamed Video and Video Conferencing 340

Further Information 344

Exercises 345

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 8
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

11 Animation **348**

Captured Animation and Image Sequences 351

'Digital Cel' and Sprite Animation 356

Key Frame Animation 358

3-D Animation 367

Further Information 374

Exercises 374

12 Sound **376**

The Nature of Sound 377

Digitizing Sound 383

Processing Sound 388

Compression 396

Formats 402

MIDI 404

Combining Sound and Picture 411

Further Information 413

Exercises 413

13 Combining Media **415**

Hypermedia 417

Synchronization-Based Presentations 425

Accessibility 442

Further Information 445

Exercises 445

14 Events, Scripts and Interactivity **447**

Scripting Fundamentals 449

World Wide Web Client-Side Scripting 450

Behaviours 466

Timeline Scripting and Behaviours 472

Beyond Scripting 483

Further Information 490

Exercises 490

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 9
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

15 Multimedia and Networks **492**

Protocols 494

Network and Transport Protocols 495

Multicasting 500

Application Protocols for Multimedia 502

Quality of Service 513

Server-side Computation 516

Further Information 522

Exercises 522

Multimedia Practice **524**

The Projects 529

Afterword by Brent MacGregor **542**

Annotated Bibliography **548**

Index **560**

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 10
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

This page intentionally left blank

Preface

This book is a text for introductory courses on digital multimedia. There is at present a diversity of approaches to such courses, ranging from the purely technical, with an emphasis on underlying system technologies or multimedia programming, to those aimed primarily at multimedia content production. Increasingly, courses with a strong practical orientation are being offered within computing and IT departments, to students coming from very mixed backgrounds who are aiming at a career in multimedia. The book will be of use to students on all of these types of course. For multimedia producers, we describe the way media are represented and processed digitally and the consequences that follow. For software engineers, we describe what multimedia producers do with the data and why they want to do it. Ultimately, these are two sides of the same coin, and an understanding of both sides will be of value to anyone seeking to work professionally in multimedia.

In order to focus tightly on these concerns, and to keep the size of the book within reasonable bounds, we have only dealt lightly with some topics. We have not described multimedia design methodologies or multimedia production management: one of the things we do when we're not writing books is explore — in our own work and with students — the potential of digital multimedia for new forms of expression that are enabled by digital computers and networks. It is clear that this potential has barely begun to be properly understood or exploited. To claim to offer design methods for multimedia at such an early stage in its evolution would be arrogant. Instead, in our final chapter, we invite readers to explore the problems and possibilities through a wide variety of multimedia projects, which are intended to provide some insight into the forms and processes of multimedia through practical experience.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 12
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Concerns about human computer interaction and ethical issues can be found throughout the book, but we have not presented a formal or thorough treatment of either; nor, since this is an introductory text, have we gone into the utmost detail about system issues, although our description of technology is rigorous and accurate. All of these topics can be pursued in depth in more specialized books, to which we include references.

In order to fully appreciate the problems and potential of multimedia, it is essential to have a good understanding of the characteristics of the media that are being combined. Armed with such an understanding, much of what happens when media are brought together becomes clear. In our descriptions of graphics, text and time-based media, we do not lose sight of our ultimate subject of multimedia, and have emphasized the relevant aspects, while eliding some topics that would normally be central to the study of these individual media as subjects in their own right.

To write about multimedia production without making reference to the specific software tools that are used in the industry would be artificial and lend an air of unreality to our descriptions. Accordingly, we describe particular applications when we need to explain how images, video, and so on, are produced and manipulated. We have chosen to use only packages that are widely used in the real world, and that students might be expected to encounter after graduation, if not before. Access to the programs described in the text, or their equivalent, will be a considerable advantage to students. A complementary volume, entitled *Digital Media Tools*, which deals with a selection of the relevant programs in more detail, is in preparation as this book is going to press.

Our treatment of multimedia is somewhat biased towards the World Wide Web. We have featured the Web, rather than more advanced forms of distributed multimedia, because students are likely to have ready access to the Web itself and to tools for producing Web content; because the emphasis on textual representations and markup languages makes it easy to see and explain what is going on; and because of the ready availability of standards and other documentation. Additionally, although we are reluctant to make predictions about the future, it is clear that the Internet is going to become a major distribution medium for multimedia content, as shown, for example, by the speed with which leading software manufacturers are adding Web features to their page layout, graphics and multimedia applications, and developing new tools specifically for Web page development. Many graduates of multimedia courses will be hoping to pursue careers in this area.

Towards the end of the book, readers will find an afterword by Professor Brent MacGregor, head of the School of Visual Communication at Edinburgh College of Art. This short essay considers what may, or may not, be the future of multimedia, and following on from the final chapter, effectively throws down the gauntlet to all those who will be working in multimedia in the future.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 13
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

We make some assumptions about our readers' prior knowledge. They should have experience of using a modern graphical user interface to an operating system such as the MacOS or Windows, and should have used the World Wide Web and seen other forms of multimedia. We do not assume any mathematical or computer science knowledge, except in specifically flagged passages (see below), but some programming experience will be valuable in understanding Chapter 14.

Presentation

Some aspects of the presentation require comment.

⇨ Some passages are typeset like this one, in a smaller font, preceded by a ⇨ symbol. These passages are 'asides' — they contain material that is somewhat tangential to the main presentation and may be omitted without great loss.

➲ Other passages are typeset like asides but in preceded by this ➲ symbol. These contain material at a higher technical level than the surrounding passages. They might be found difficult by readers with little technical and mathematical background, but should be illuminating to those with such knowledge.

The Web Site

Teaching about multimedia by writing is rather like teaching swimming on dry land. An extensive Web site, with examples of those media that cannot be appreciated by reading alone, expands the limited illustration of material which is possible in a printed book. Additional supporting material, including links to other Web sites demonstrating multimedia, are available from the site, together with product reviews, and hints on solutions for exercises. The URL for the site is `http://www.wiley.com/digital_multimedia`.

Acknowledgements

We are grateful to Adobe Systems, Inc., Electrifier, Inc., Macromedia, Inc., MetaCreations Corp. and Terran Interactive, Inc. for their generous support of this project. We would also like to thank David Barron, University of Southampton; George Beekman, Oregon State University; Rhonda Berger, Miami Dade Community College; Andrew Calway, University of Bristol; Russell Campion, University of Staffordshire; Maxine Cohen, Nova Southeastern University; Greg Cranitch, Griffith University; Stephan Fischer, Darmstadt University of Technology; Rachelle Heller, George Washington University; David Hutchison, University of Lancaster; Peter Kenny,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 14
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

University of Kent and Jimmy L Reeves, Texas Tech University, all of whom read and commented on a draft of the book. We are also grateful to John Taylor for help with Bryce.

We have received wonderful support — better than we deserved — from our publisher John Wiley & Sons, and we would particularly like to express our thanks to Dawn Booth, for supervising the production, Katrina Arens, for expediting things and making a great many telephone calls, and above all, to Gaynor Redvers-Mutton, for her encouragement, enthusiasm, patience, and for one or two small miracles.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 15
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Introduction 1

It was a dark and stormy night.

A storyteller might use these portentous words to capture your attention and draw you into a tale. A novelist might use the same words to begin the first page of a book. A radio play based on the same story could open with a crash of thunder and the howling of the wind. A film of the book could open on a scene of darkness, suddenly illuminated by a flash of lightning that reveals trees bent over in the gale.

The dark and stormy night can be represented in different *media*, each of which tells the story through different means, appealing to different senses. One of the key insights in computing is that all these media can be represented *digitally*, as a structured collection of bits, and can then be manipulated by programs on a computer, stored on disks and other storage devices, and transmitted over networks. Their shared digital representation means that different media can be combined into what is loosely called *multimedia*.

The combination of different media is not something new. On the contrary, established forms mix media routinely. A TV news bulletin, for example, might include sound and moving pictures — both live and recorded — still images, such as photographs of politicians, graphic illustrations, such as a histogram showing the trend in unemployment, and text, in the form of subtitles, captions, quotations, and credits, which will usually be accompanied

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 16
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

by suitable annunciatory music. A contemporary live theatrical performance may incorporate two- or three-dimensional artwork and design (sets, backdrops, costumes), sound, which may be a mixture of speech and live music, perhaps with tape recordings and synthesized sounds, projected still and moving images, and sometimes text in the form of surtitles (written text displayed above the stage, which are often used for translation, especially in opera).

The integration of media is natural — we perceive the world through all the senses we have at once; the separation of media is artificial, and may often seem unsatisfactory.

Consider film, for example. From the earliest days of cinema, in the absence of any method of recording and playing sound in synchronization with picture, separate live sounds were produced at the same time as the film was shown: not just the live piano accompaniment that you may find in contemporary art-house cinema showings of silent classics, but dialogue, singing, and sound effects, produced by actors, singers, and noisemakers in the theatre, often hidden behind the screen. So great was the film-going public's desire to hear sound naturally integrated with moving pictures that in 1927 when Al Jolson spoke his phrase, 'Wait a minute, wait a minute. You ain't heard nothing yet,' in *The Jazz Singer*, audiences stood up and cheered. It is hardly surprising, then, that once the technical obstacles to representing media digitally (which we will consider in Chapter 2) were overcome, the next step should be to combine separate media. In fact, the separation of digital media into distinct data types — different 'sorts of things' to which different operations can be applied — can be seen as a temporary exigency demanded by an immature technology.

Applications of Multimedia

What is it, then, if not the combination of media, that distinguishes digital multimedia from previous forms of combined media? It is the fact that the bits that represent text, sound, pictures, and so on can be treated as data by computer programs. The full potential of this fact has not yet been fully explored, but one facet of it immediately distinguishes digital multimedia from its predecessors. A program can control the order in which various components are presented and combined, and can do so in response to input

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 17
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

from a computer user. In other words, digital multimedia can be *interactive*, in a way that, for example, a TV news bulletin is not, and that goes far beyond the simple control afforded by a VCR.

The digital multimedia version of the story we began this book with could open with a picture depicting a dark and stormy night. By clicking on an icon on the screen, the user could cause the scene to play as a video clip, or add sound effects, according to their choice. Perhaps by clicking on different areas of the scene, details could be revealed, or a new scene could be introduced — a different one for each part of the first scene, allowing the tangled tale of betrayal and redemption that began on that dark stormy night to unfold in more than one way, told through the eyes of different characters. Users whose hearing was poor could choose to have a transcription of any dialogue displayed for them, while users with impaired vision could have a description of what was happening 'read' to them by their computer. Different interface options might be offered for users with different tastes and needs. If the story had attracted critical attention, it might be possible to bring up learned commentaries on particular aspects, or a history of its telling in different media. There might be an opportunity for the user to take an active rôle, change the story, add new elements or recombine the existing ones.

Suppose, though, that this is not just a story — that some crime was committed on a real dark and stormy night, and that somebody had subsequently been arrested and charged with an offence. By collating evidence from witnesses, and facts about the location, the events of the night could be reconstructed and presented, like our story, as a mixture of animation, graphics, sound and video, with textual annotations. Members of the jury could then examine the event from different angles, verify that it was possible to see something at a certain distance, perhaps factor out the parts of the reconstruction based on a certain witness's evidence, or try out various hypotheses suggested by the lawyers in the case. *Forensic multimedia* of this sort is already being used in courts in the United States.

Digital multimedia may be used for many things besides stories and forensic reconstructions. The most prominent and economically significant uses of multimedia at present are in the fields of entertainment and education. At one extreme, the combination of sound, animation, video and text with interactivity is commonly found both in adventure games, such as *Starship Titanic* and action games, such as *Doom* and its bloody offspring. At the opposite

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 18
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

extreme are the multimedia reference works, such as the *Cinemania* cinema encyclopedia, or the digital versions of the *Encyclopedia Britannica.*

Multimedia is also being used educationally in schools, and for extra-curricular and pre-school learning, where multimedia teaching materials can be used to present information in more direct ways than traditional books and lessons. For example, film clips and original recordings of radio broadcasts can be integrated with a textual account of events leading up to the Second World War in a modern history lesson. Virtual science experiments can be constructed using multimedia techniques, to allow pupils to work with simulations of equipment that is not available to them. In both these cases, interactive features allow pupils to work at their own pace, or to pursue specialized aspects of a project according to their particular interests. Multimedia teaching materials can be bought in as complete CD-ROM titles, or they can be made by teachers in their own schools. Children themselves can also make multimedia, in the form of project reports, or as an end in itself, as an electronic class book or newsletter.

The distinction between education and entertainment is not a clear one; several succesful multimedia CD-ROMs use elements of games for an educational purpose. For example, *The Logical Journey of the Zoombinis* develops children's problem-solving skills through a series of puzzles, organized in the form of a game. This approach to learning seems to be popular for very young children, with a range of titles available for teaching literacy and basic mathematics. These are aimed both at schools and at parents, for use in the home. (As a side-effect, the use of such materials also teaches computer skills from an early age.)

In contrast to the entertainment and education sectors, where most multimedia is professionally produced and marketed, much of the material used in business is produced for a specific company, often in-house. In a continuation of the use of multimedia in formal education, multimedia materials are used for training. As with the virtual science experiments in schools, multimedia can be used to produce interactive simulations of equipment and procedures, in cases where it would be expensive or hazardous to use the real thing for training. Multimedia is also used in sales presentations and promotions — for example, computers on sale in showrooms often run multimedia presentations describing their capabilities, and Web pages often function as sales brochures.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 19
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Some professions and business areas lend themselves especially well to multimedia. Graphic designers produce interactive presentations of their work as a supplement to, or even as a substitute for, a conventional portfolio. The electronic portfolio can be inexpensively duplicated on CD-ROM and sent to potential clients, or posted on a Web site as a virtual exhibition. In the shrink-wrapped software business, paper manuals have been gradually giving way to electronic documentation for some years. The move from hard copy to digital formats has provided an opportunity to augment the text and diagrams of conventional manuals with animations, video clips, and sound. Many programs now come with a multimedia product tour, demonstrating their best features, and interactive tutorials, which guide novice users through simple tasks.

One area in which computation plays a more active part is *visualization*, in which graphics and animation are used as a means of presenting complex data. Banal examples of visualizations include pie-charts and bar-charts being generated from spreadsheet data: their graphical form summarizes the data in a way that makes recognition of trends and comparisons between magnitudes possible at a glance. More complex data than the sort normally kept in spreadsheets demands more complex visualizations to provide the same degree of comprehensibility. Three-dimensional and time-varying presentations, making use of false colour to carry extra information, are often favoured. Such elaborate visualizations are frequently generated from simulations of complex dynamic systems: for example, a program that simulates atmospheric dynamics may generate a time-lapse movie of the gestation of a tropical storm; a program simulating an organic chemical reaction may generate a three-dimensional model of the structure of the molecules involved, showing the changes they undergo. As these examples show, visualization is widely used by scientists; it is also used in business, where the systems being tracked or simulated are financial, and the output is correspondingly more abstract.

The established entertainment industry, while still delivering an essentially linear and non-interactive product, now makes extensive use of digital multimedia technology in the production of everything from blockbuster movies to gardening programmes on TV. The use of connections with digitally delivered material, such as links between programming and supporting Web sites, or accompanying CD-ROMs for mainstream movies, is rapidly expanding. At the same time, a rapidly increasing number of festivals, conferences and

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 20
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

other international events are providing showcases for "new media" at all levels from the artistic avant garde to the purely commercial.

Delivery of Multimedia

The applications we have mentioned so far are all basically concerned with presenting material, using a combination of media, in an interactive way. Some means of delivery is required, to get this material from its producer to its consumers. It is useful to distinguish between *online* and *offline* delivery.

Online delivery uses a network to send information from one computer, often a server machine providing centralized storage of bulky data, to another, usually a personal computer on somebody's desk. The network in question may be a local area network serving a single organization, but more often it will be the Internet. In particular, the World Wide Web has established itself as a popular means of delivering multimedia online.

⇨ The World Wide Web has achieved such supremacy as a means of accessing data over the Internet that in popular perception the Web and the net are sometimes seen as synonymous. This is not the case, as we will explain in Chapter 2, and in more detail in Chapter 15.

Where multimedia is delivered offline, some removable storage medium must be used. The widespread deployment of CD-ROM drives in personal computers in the mid-1990s was, to some extent, responsible for the surge in interest in multimedia at that time. This is because, for the first time, it made available a form of storage with adequate capacity (around 650 Mbytes, compared to the 1.44 Mbytes of floppy disks) to accommodate the large files characteristic of multimedia data, while still being relatively inexpensive to duplicate. The data transfer rate of early CD-ROM drives was too slow to support playback of video and sound, but more recent models have increased the speed, to the point where multimedia can be played directly from a CD-ROM, although smooth playback of full-screen video from CD-ROM is still impossible without additional hardware support.

While online delivery is widely seen as the future for multimedia, the low bandwidth available for most domestic connections to the Internet means that, especially for large and elaborate work, offline

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 21
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

delivery on CD-ROM remains popular. There is an increasing trend towards a combination of offline and online delivery, with CD-ROMs being used to hold a fixed collection of data, augmented with links to Web sites, where updates and supporting information are available (sometimes for a supplementary fee).

In 1995, an industry consortium announced the specification for a successor to CD-ROM, using the same size and shape platters, called DVD. Originally, this stood for Digital Video Disk, since the format was intended as a replacement for VHS cassettes as a distribution medium for video, as analogue methods of storing video began to be replaced by digital methods. It was immediately understood that DVD could be used equally well for any digital data, just as the audio Compact Disc had been adapted as the CD-ROM. Accordingly, the name was changed, somewhat desperately, to Digital Versatile Disk, thus keeping the abbreviation DVD. DVD offers a much higher storage capacity (up to 17 Gbytes on a double-sided disk), with similar transfer rates to modern (12× and higher) CD-ROM drives. Despite considerable media enthusiasm for DVD, it has not been taken up as rapidly as originally predicted, but during 1999, an increasing number of personal computers were being shipped with DVD drives fitted. Since these drives can also read CD-ROMs, it seems likely that they will become standard peripherals, and that DVD will be used to deliver multimedia, especially high quality video.

Online delivery, however, offers possibilities which are not available offline. In particular, it enables the delivery of (almost) live multimedia content, which in turn makes possible novel applications such as video conferencing and broadcast multimedia. Generally, when multimedia is delivered online, the delivery need not be passive, in the way that it is from CD-ROM: things can happen at the delivering end; for example, a database query can be executed.

Historical Context

Compared with established forms such as film or the novel, multimedia is still in its earliest days. The CD-ROM specification was published in 1985, with drives appearing on desktop machines from about 1989; the World Wide Web became publicly available outside CERN at the start of 1992, in the form of a line-based

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 22
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

browser giving access to a handful of servers; by January 1997, when the HTML 3.2 specification was adopted as a World Wide Web Consortium Recommendation, audio and video were only supported in Web pages through the use of proprietary extensions. We have become so accustomed to the high speed of technological change in computing that multimedia is already seen as an established feature of the computing landscape. But, on the time-scale of cultural change, it has barely arrived.

The history of film and animation, for example, demonstrates that it takes time — much longer than digital multimedia has existed — for conventions about content and consumption to become established. The very first films exhibited to the public, by the Lumière brothers at the Grand Café on the Boulevard des Capucines, in Paris in 1895, showed such subjects as workers going home from a factory, and a train arriving at a station. To be able to reproduce movement was enough, without the farrago of plot, character, or message we normally associate with films. The early trick films of Georges Mélies were shown as part of his magic show, in the same way as the magic lantern projections he used to create illusions. In a similar vein, Winsor McCay's *Gertie the Dinosaur*, one of the first short animations made in America (in 1909), was used as an integral part of his vaudeville act. While Gertie did tricks, McCay stood in front of the screen on stage and talked to her, telling her what to do, and scolding her when she got it wrong[1], in response to which Gertie started to cry; finally McCay appeared to walk into the frame and ride off on the dinosaur's back.

1
In some versions of the film currently in circulation, this monologue has been added as a soundtrack, but the disembodied voice does not convey the presence or interactive nature of the vaudeville act.

At the same time, films, including narrative and animation, were already being shown, particularly in Europe, as entertainments in their own right, purely to be watched on a screen. The diversity of early ways of using film shows that there was originally no consensus about a 'right' way to present a film. Later on, anything other than a cinema screening would be seen as eccentric. Even films that attempted to use the medium in original ways would mainly do so within the cinema context, and a film shown in other ways, for example projected onto the floor of a gallery, would be re-defined — for example, as an 'installation'.

Another notable feature of early cinema is the way in which established forms were translated into the new medium. In particular, the newsreels shown around the time of the First World War were based on the same principles as a newspaper, even to the extent of including an animated cartoon corresponding to a comic

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 23
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

strip. Characters were transported from the comic strips to film, and the animators who brought them to life were often (but not always) the artists who had drawn the strips. Perhaps more to the point, one of the most succesful studios producing newsreels and animations was the International Film Service, owned by the newspaper proprietor William Randolph Hearst — who also owned the syndication rights to many popular comic strips of the period.

Remember that the time we are now describing was twenty years after the invention of cinema, yet we still find film looking to earlier media for its forms and content. In a similar way, multimedia still adopts the format of earlier media; the most obvious example is the multimedia encyclopedia, which has the same form — a series of short articles, accessible through one or more indexes — as its paper equivalent, and is often published by the same organization. Other reference works also follow the form of reference books, sometimes, as in the case of some of Dorling-Kindersley's CD-ROMs, following the distinctive house style of their originals. Electronic software manuals rarely depart far from the organization of hard-copy manuals, merely adding better searching facilities and sometimes some animation.

One of the things that was needed before film could acquire its distinct character as a medium was an appreciation of the way the movie camera and film could be used to create new sorts of images, by framing, movement, editing and special effects. In multimedia, part of what is presently missing is a real understanding of how we can take advantage of the fact that digital multimedia is data, to integrate the presentation of multimedia with computation. For the moment, we are largely confined to controlling the presentation interactively, and to moving the data across networks.

Multimedia Production

The making of multimedia requires software not only for the preparation of individual media elements, but for their integration into a finished production. Programs that allow a designer to assemble different media elements in space and time, and add interactive behaviour to them, are usually called *authoring systems*. These must interpret user actions and commands in non-trivial ways in order to translate the mode of working that is natural to a designer, who

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 24
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

will usually feel most comfortable with direct manipulation of the elements on a virtual pasteboard or its equivalent, into the data and control structures of the finished piece of work.

Multimedia authoring systems take many forms, though: if we are currently unsure about the form of multimedia, we are equally uncertain about the best means of producing it. Again taking film for comparison, we find a well-defined set of jobs, from director to clapper operator, and established procedures for coordinating the stages of production that are involved in the making of a finished film. No equivalent division of labour or procedures have yet been established for multimedia. Some authoring systems, such as Director, attempt to provide tools for the production of all types of media element as well as for their integration, almost as if a single person could do the whole job, working on different parts at the same time. Other tools, such as Photoshop, only work with one type, suggesting a division of labour along the lines of the different media, but requiring separate mechanisms and a separate production stage for integration.

Just as the forms of multimedia are largely derived from the forms of other media, so attempts to impose order on the process of making multimedia are largely based on procedures from other media. *Storyboards*, for example, are sometimes advocated as a means of making and communicating multimedia designs. Storyboards are used in large animation studios, and by music video and advertisement producers, to plan the construction of a piece of work, and to communicate its structure among the team producing it. (In the industries where storyboards are widely used, they are also a means of selling ideas to financial backers.) A storyboard is simply a sequence of still pictures, showing the composition of shots at key points in the production. Rather like a blueprint, it records a plan of the work, which can be used to organize its subsequent development. One of the things that makes storyboards effective is the way they can be read linearly like a comic strip. As we will see in later chapters, and as you probably know from your own experience of Web pages and CD-ROMs, multimedia is often arranged non-linearly, with the possibility of taking branches, so that the elements may be displayed in different orders. When non-linearity is added to a storyboard, by annotating it with arrows, for example, the clarity with which it communicates structure, and thus its *raison d'être*, is lost. Conversely, starting out with the fundamentally linear format of a storyboard when planning

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 25
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

multimedia will inevitably prejudice the designer in favour of linear structures.

➲ A parallel can be drawn with the use of flowcharts for program design. The conventions of 'classical' flowcharts can only represent conditional and unconditional branches. A flowchart design therefore suggests an implementation in terms of jumps, not in terms of the higher level control structures supported by modern programming languages. The 'structured programming' movement of the 1970s discarded flowcharts, in favour of working directly with programming language constructs that express control abstractions. It might be hoped that multimedia authoring tools might themselves take on a design rôle, in a similar fashion.

Terminology

A telling symptom of the immaturity of digital multimedia is the absence of satisfactory terminology. What do you call a mixture of media under software control? Where the display and presentation of the media elements is the sole purpose, as in a Web page or an encyclopedia on CD-ROM, we will usually refer to a *multimedia production*. Where the display of the multimedia is more intimately bound up with computation, we will refer to a *multimedia application*. A simple example of a multimedia production might be a Web page containing a chronological listing of forthcoming film festivals, with some attached images and video clips. A multimedia application using the same data could provide an interface (which might still be a Web page) to a database containing the festivals' details and related material, so that a user could search for festivals in a particular geographical area, or those that accepted material on video tape, and so on. The distinction between the two is not a sharp one. A certain amount of computation might be done to control the presentation of a multimedia production, so there is a continuum from productions that simply present material in a fixed way, through increasingly complex interactivity, to applications that generate multimedia dynamically.

There remains another question. We know that we *read* text, *look at* images, *watch* video and *listen to* sound, but what do we do to a multimedia production? All of these, perhaps, as well as interacting with it. There is no satisfactory term either for the act

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 26
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

or for the person who performs it. For the latter, we reluctantly fall back on the over-worked and vague term *user*. The usage at least has the virtue that it draws attention to the active rôle that, potentially anyway, a user adopts with relation to most multimedia productions. Although much influential critical work has focused on the active rôle of the reader in creating readings of texts, this activity is of a cerebral and largely unconscious type. In contrast, the user of multimedia is active in a literal sense, using input devices such as the mouse and keyboard to control what goes on — within the limits allowed by the multimedia producer.

It can be useful to distinguish between *multimedia* and *multiple media*. The distinction is best understood from the point of view of the user. It is commonplace that we perceive different media in different ways: we just referred to reading text, looking at pictures, listening to sound, and so on. Cognitive scientists call these different sorts of perception *modalities*. A multiple media production requires us to switch between modalities, for example, by reading some text, then looking at a picture, and so on. Many early 'multimedia' CD-ROM presentations were like this. True multimedia requires us to combine modalities (as we do in real life — think of the experience of walking through a shopping mall, for example), or even develop new ones, to cope with the combination of media. A familiar example is the pop video, where the music and moving images are (usually) presented as a composite — arguably often to the detriment of one or the other considered in isolation — with, for example, visual action illustrating lyrics, and scenes being cut in time to the beat of the music.

While modalities provide a useful way of thinking about multimedia from a cognitive point of view, we prefer to adopt a definition more in keeping with the technical emphasis of this book, and consider digital multimedia to be

> any combination of two or more media, represented in a digital form, sufficiently well integrated to be presented via a single interface, or manipulated by a single computer program.

This definition is broader than some that have been proposed. It is quite common to require at least one of the media to be a time-based one, such as sound or video. It is certainly true that, at present, the addition of time introduces significant new technical difficulties, which may be considered sufficiently challenging to

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 27
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

make a qualitative difference between such dynamic multimedia and purely static multimedia, involving only text and images. However, we consider these difficulties to be transient effects of the current state of the underlying digital technologies, and prefer to use the more general definition just given.

Interactivity

"You can mistake endless choice for freedom"
Bruce Springsteen, speaking on BBC2, December 1998

He was talking about something more momentous than multimedia productions, of course, but his words are curiously appropriate to the way in which interactivity has been applied to multimedia.

Interactivity is frequently portrayed as the feature that distinguishes digital multimedia from other forms of combined media, such as television. It is often claimed that it puts the user in control — by implication, in a way that has not previously been possible: "Interactivity empowers the end users of your project by letting them control the content and flow of information."[2] While there is some truth in this, it should be recognized that the amount of control offered is strictly limited within parameters established by the multimedia producer. This must be the case in any situation where interaction takes place with a finite system responding according to a program. Only choices that are coded into the program are allowed. However, providing choices in a computer program is much easier than providing choices via a hardware device — such as through the controls of a VCR — so it is possible in a multimedia production to allow the user to control events at many points. Where a sequence of choices can be made in succession, the possibilities expand combinatorially: if, for example, there are four choices at each of five stages of a production, although only twenty branches must be programmed, there is a total of $4^5 = 1024$ possible sequences which can be followed; that is, 1024 possible ways of experiencing the production. Such a range of possible experience might well be mistaken for freedom, but ultimate control over "the content and flow of information" remains with the producer not the user, and even a significantly increased range of choices will not necessarily enhance a production. In certain cases — e.g. the provision of access to information in

2 Tay Vaughan [Vau98].

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 28
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

facilities such as Web site railway timetables — a single, fast, optimized route to the required information is all that is needed. And no amount of interactivity can compensate for poor or dull content, or bad organization of the work — one of the worst cases of this is when, through a series of selections on buttons, etc., one navigates through a site to find that the desired page is an empty space labelled "under construction".

The character of interaction can be appreciated by considering how it operates in the popular game *Myst*. The player is presented with a series of images, depicting scenes in an imaginary world. Sometimes, if you click on an object, something happens — doors open, machines operate, and so on. When the cursor is placed near the edge of the screen, it changes shape and a mouse click causes the scene to change, as if you had walked out of the frame into the adjoining space; in this way you can, for example, walk round a building. Apart from that, there are no controls, such as buttons, so the game demands a process of exploration before you can find out how to make anything happen. Advertising copy and reviews of the game placed great emphasis on the idea that you have to explore a fantastic world. In fact, though, all that your choices and explorations enable you to do is to complete a few little puzzles, obtaining fragments of a story along the way, until you reach the end. You don't have any choice over this — if you can't work out the combination to open a door, you can't attack it with a crowbar instead, for example. All you can do is go off and try a different puzzle, or give up. However, the extraordinary popularity of *Myst* demonstrates how interactivity, when embedded in a rich environment with high-quality graphics and evocative sound, can dramatically increase the attractiveness of a production. It is hard to believe that the puzzles presented on their own, or the flimsy piece of fantasy fiction upon which the game is built, would draw the same audience.

⇨ 'Interactivity' is really a misnomer — although we will continue to use the term in deference to its wide currency. When the computer's rôle is to present choices and respond to them, it cannot be said to be keeping up its end of an interaction, while at the same time it reduces the user's options for contributing to the intercourse to a few mouse gestures. The application of artificial intelligence can improve the level of discourse, but (with all due respect to Alan Turing) true interaction is only possible where another person is involved. Even then, if the interaction is mediated by a program, the form it can take may be severely limited. In many contemporary

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 29
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

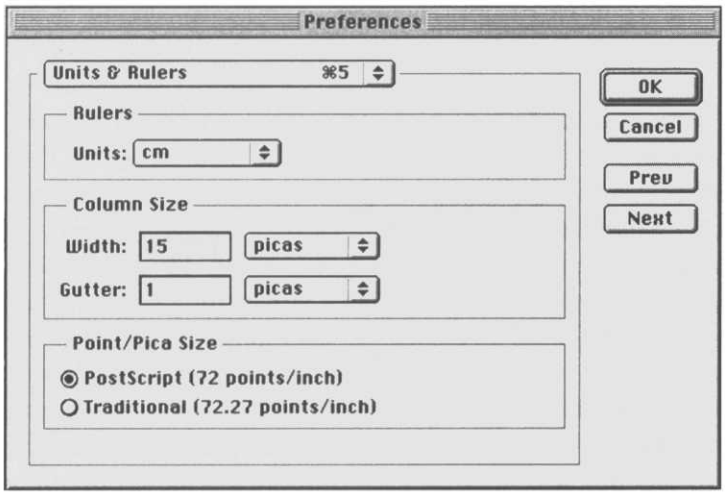


Figure 1.1
A typical dialogue box

networked games, for example, the only way you can interact with your fellow players is apparently by trying to kill them.

Since interactivity is a novel contribution that computers make to multimedia, there is a strong tendency for all multimedia productions to be interactive. It is worth remembering, though, that 'endless choice' may not only be inappropriate or redundant in the ways described above — it is not necessarily what everybody always wants. In some cases an audience may be quite content to let a story unfold just the way the writer or director intended it to, without the distraction of commentaries, out-takes, or supplementary material. In many of these cases, it would probably be better to read a book or go to the pictures — digital multimedia is not appropriate for everything. There will also be times when producers do not wish to offer interactivity to the user. An increasing number of artists are creating work in the form of digital multimedia, but in many cases they choose to retain control of what is shown, and do not allow their audience to interact with the work. Such productions can be innovative, and even seek to push the boundaries of what can be achieved in the digital domain, without being interactive or departing from an essentially linear form of presentation.[3]

3 Perhaps in the future some digital multimedia productions may provide a linear experience that is nevertheless unpredictable, because the development at each viewing could be controlled by computation.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 30
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

User Interfaces

The means by which choices can be presented to users can vary enormously. At one extreme there is the stylized set of user interface elements — menus, dialogue boxes, outlined buttons, and so on — and conventions used by most operating systems and mainstream applications; at the other extreme, interaction with some sorts of game is essentially free-form, with any part of the screen liable to cause a response at some time. Adoption of the conventions has the advantage of predictability: users will generally know what to do when confronted with a dialogue box, such as the one shown in Figure 1.1, containing text entry areas, pop-up menus and buttons — fill in the text, make a selection from the menus, click on the buttons. By following the interface guidelines for the platform your multimedia production is intended for, and designing your dialogues and other interface elements carefully, you can often make it possible for an experienced user of that platform to work with your production without further instruction. A disadvantage is that users who are not already familiar with the interface conventions might find it harder to use a fully-fledged conventional interface than a specially designed simpler one. Another disadvantage is that everything looks the same; you might have legitimate reasons for wanting to differentiate the appearance of your interface from the operating system. Doing this while at the same time making it clear how your various controls work is a difficult task. The innovative interfaces designed by Kai Krause for products such as KPT and Bryce (see Plate 1[4] and Figure 4.49 in Chapter 4), for example, have sharply divided users, with some hailing them as a breakthrough while others criticize them for being confusing and affected.

4
By default, Bryce's interface takes over the whole screen, obscuring the desktop entirely and hiding the menu bar the way some games do. This is in direct contravention to at least the MacOS user interface guidelines — it's actually quite difficult to do on a Mac — but it doesn't prevent the program from being usable, at least in many users' opinion.

The interface elements that have become most familiar to desktop computer users were devised in the context of a static graphical environment. With the incorporation of time-based media — video, animation, sound — new types of operation, such as starting, pausing, rewinding, and adjusting the volume, become possible, and so new types of control are required. Menu commands with keyboard shortcuts could be used, but the practice of supplying playback controls in the form of buttons that mimic the operation of those on cassette players and VCRs, and are labelled with the *de facto* standard icons used on such appliances, is more common. Volume controls sometimes have the appearance of sliders or

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 31
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

thumb wheels. Since digital video players make it easy to go to a specific point or scrub through a movie, using mouse movements, an additional control is often supplied in the form of a strip, representing the movie, with a marker that can be pulled through it; this marker moves during normal playback to show how much of the movie has been played (see Figure 1.2). The requirements of incorporating time-based media have thus given rise to new interface conventions. As always, these conventions can be ignored, and a free-form or individualistic interface substituted where that is felt to be justified by the nature of a particular production. It should be borne in mind, however, that accepted interface designs are the tried and tested products of expert designers, and the attempt to create a radical or superior interface should not be undertaken too lightly.

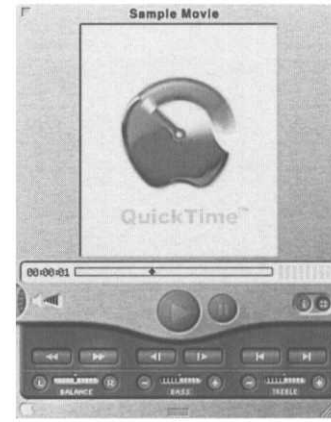


Figure 1.2
Controls for playing video

Social and Ethical Considerations

It is a commonplace among technologists that technology itself is neutral and without moral value, either good or bad. It is only the uses to which people choose to put technology that need to be subjected to moral scrutiny. While this assertion is literally true — because ethics by definition applies only to the thoughts and actions of sentient beings — it is often the case that, in a specific social, political and economic context, the introduction of new technology presents opportunities for behaviour that were not there before. In such cases, certain ethical problems caused by that behaviour can conveniently be described as arising from particular technological innovations.

Access to Multimedia: Consumption

In the case of multimedia, the issue springing most directly from the nature of the technology itself is that of access. To begin with, access to multimedia depends on access to appropriate hardware and on possession of the necessary skills to use a modern computer system. In developed countries, access to hardware is largely a function of wealth. Although personal computers have become a consumer commodity, they remain among the more expensive of such items: a personal computer equipped to handle multimedia

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 32
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

costs substantially more than a standard VCR or a games console, for example. In less developed countries, computers remain rare, and the infrastructure to support modern computer systems — reliable power supplies, ubiquitous telecommunications — is absent. In some countries large numbers of people are still denied the opportunity to acquire basic literacy — this, too, is often a side-effect of poverty. It is sobering for the multimedia developer to remember that substantial numbers of people around the world live without even those basic domestic facilities and appliances which members of the more privileged societies take for granted — in conditions where issues such as access to clean water are of vital importance, and access to computing not even in question. Any talk of the global nature of the Internet, or of multimedia distribution, should be tempered by this knowledge.

Even among the wealthiest countries, local conditions can affect the ease with which certain kinds of multimedia can be accessed. In most parts of the United States, for example, a single charge covers the cost of all local telephone calls, effectively making individual calls free. It is thus quite feasible to leave a personal computer permanently connected to the Internet over a telephone line. In the United Kingdom, local telephone calls are charged for by the second, so most people with dial-up Internet connections try to minimize the time they are online. Somebody in Britain is much less likely to download huge video files, or tune in to a live Web audio broadcast, than their counterpart in the U.S. The Internet is a global phenomenon in principle, but in practice it wears different faces in different places, and in many places it has no face at all.

There are other factors which can prevent access to multimedia. As we stated earlier, the ability to use a computer system is a prerequisite — and this in itself presupposes basic literacy. These skills are by no means universal, even in developed countries. There is conflicting evidence about how computer skills are distributed among age groups. Certainly, young people in most developed countries are now routinely exposed to computers in schools, and often at home, if only through games. But the skills they acquire may be superficial, and the quality of computer education varies as much between schools as any other subject does. The highest levels of computer skills are usually acquired through work, so a failure to master computer technology may be an increasingly significant effect of long-term unemployment. And, of course, not all forms of employment expose people to computers, so disparities of skill

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 33
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

are found between occupations, which may, in time, reinforce other social distinctions — e.g. between office and manual workers. People who are too old to have been exposed to computers in schools and who have not used computers as part of their work may find even a user-friendly graphical interface perplexing, so, even if they can afford the hardware to access multimedia, they may not feel able to take advantage of it.

Finally, physical disabilities or learning difficulties may interfere with a person's ability to use computers or experience multimedia content. This is particularly poignant as computers can be of great value in ameliorating some of these difficulties. A range of problems, from arthritis to motor neurone disease, may make it difficult or impossible for someone to operate a conventional keyboard, mouse and other input devices. Blindness and visual impairment raise specific concerns. Voice synthesizers and Braille keyboards and output devices can make it possible for blind people to use computers effectively (again assuming sufficient wealth or social provision makes such devices available), but most multimedia has a marked graphic bias, relying on images not just to convey information, but for navigation and interaction. This is evident on the World Wide Web, where many Web pages use image maps for navigation; the habit of using small images as navigation icons (or merely to provide text labels in a more interesting font than the defaults) means that some Web pages are unusable without images.[5] People who are dyslexic, on the other hand, typically find graphic images and icons more readily intelligible than text, and will be assisted by good design of these elements. Although it is well known that many people have difficulty in distinguishing between certain colours, and most books on design offer advice on how to choose colour combinations to minimize the problems arising from this, few designers appear to apply this advice when it conflicts with their aesthetic intent. The range of difficulties which may be experienced by potential users is very wide, and embraces all media elements, with correspondingly far-reaching implications for the conscientious multimedia designer.

The World Wide Web Consortium has done an exemplary job in addressing the problem of access for the disabled (we will describe the details in Chapter 13), but there is little evidence at present that many Web site designers are taking advantage of the facilities to make their Web pages more accessible to everyone. It is not within the scope of this book, or our own fields of expertise, to do justice

5 If you don't believe this, try turning off image loading in your Web browser.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 34
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

to this aspect of access and to provide solutions, but we stress that this is an area which requires the most careful attention.

Provision of computers in public libraries, schools, community centres and Internet cafés, inexpensive set-top boxes and network computers, popular education programmes, and accessibility initiatives can all broaden access to the Internet and other forms of multimedia. But it should be understood that access is not, and probably never will be, universal, especially outside the industrialized nations. As long as the bulk of multimedia is directed to entertainment, this denial of access cannot be considered too serious — one can, after all, live without *Riven* or *Cinemania*. Nevertheless, if access to multimedia is portrayed as the norm in a society, to be without that access becomes a marginalizing influence. When advertisements routinely feature URLs, and you have no Internet connection, what message are the advertisements conveying to you? If multimedia does indeed become the primary channel for information dissemination, as some pundits envisage, being denied access to multimedia will mean being denied access to information. On a mundane level, when information about entitlement to social security benefits, refurbishment grants or medical aid is readily available from the home to those with an Internet connection, in the form of a slickly packaged interactive multimedia presentation, but requires a trip to a crowded and depressing government office to pick up a leaflet for those without, it is easy to see how the shift of information provision to new media can deepen existing economic divisions. In extreme cases, simple access to information saves lives, as with health education programmes. Access to information is empowering, and a deliberate denial of access to information has always been one of the most powerful weapons used by governments and the rich and powerful to exercise control and protect their interests. These issues have a social and political reality, and are not simply a matter for peripheral academic debate by multimedia students and producers. The work you make, if it is released to the world at large or to a part of it, will affect other people's lives.

Access to Multimedia: Production

Anybody with access to the Internet can have their own Web site — and anybody without can't. People who cannot use the Internet for any of the reasons given in the preceding section are not only

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 35
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

denied access to the information it contains, they are also denied a platform for expression or advertisement. However, in the parts of the world and segments of society where Internet access is readily available, what computing people would call write access is almost as widespread as read access. An Internet Service Provider that did not offer several megabytes of free Web space to its customers is unlikely to survive for long. The availability of inexpensive tools for constructing Web sites (and the possibility of constructing sites by hand in a simple text editor), plus the relatively small effort needed to acquire enough knowledge to use these tools, means that anybody who feels they have something to say to the world can say it on the World Wide Web. Relatively affordable computers, together with very low cost access to the Internet and cheap CD-ROM technology, is bringing about a widespread revolution in access to the means of production and distribution of digital material.

In contrast, access to the means of production and distribution of traditional media is tightly restricted. Probably the most accessible medium is the printed book. It is possible for an author to write, print, and distribute their own books, but the costs involved are considerable, and the likely returns, in the absence of any marketing machinery, are slim. Such personal production is reserved for the independently wealthy. Most books are published through publishers, who are able to take advantage of their size to spread production costs, and to market and distribute their books more effectively. But publishers can only stay in business by making money, so they can only publish books that they think are going to sell. This means that book proposals are submitted to editorial scrutiny and refereeing — with the result that established authors have a better chance of being published than newcomers, and unconventional subject matter or styles of writing that only appeal to a minority of readers are unlikely to appear in print at all. There are small independent publishers — especially in the United States, where there is a tradition of independent and underground presses — who may pick up marginal work, but, like independent authors, their limited resources make it hard for them to achieve wide distribution. Contrast this with the Web, which offers a potentially enormous audience to anybody.

The situation in other traditional media is similar to that for books. Mass distribution of printed images takes place largely through magazines, newspapers and books, and is subject to the same editorial control as the distribution of printed text. Many

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 36
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

illustrators and photographers also sell their work to greetings cards, calendar, or postcard manufacturers, or provide images for brochures or corporate publications. Here, the influence of marketing pressures and corporate values will clearly limit the type of work that can be used. Fine artists wishing to exhibit their work are often faced with even higher barriers. Most exhibitions take place in private galleries which survive by taking large commissions on the sale of work, and can therefore not afford to take risks; public galleries, such as the National Gallery or the Museum of Modern Art, if they exhibit work by living artists at all, normally only present the work of established artists. Even if an unknown or unfashionable artist can manage to get their work exhibited, it will only reach the few people who are able to be physically present in the exhibition space. Whereas printed images can be turned into digital form and disseminated through the Internet or on CD-ROMs, the very nature of paintings and other art works means that digital dissemination can only be done at the cost of a substantial degradation of quality, and with a loss of the presence that may well be the essence of the work. However, fine artists, with their trained habits of exploring media, may be in the best position to take advantage of new opportunities that digital media and multimedia offer.

Most recorded music is distributed by a few labels, owned by huge multinational corporations, whose sales dwarf those of the independent labels. Because of the investment that a record label must make to sign, record and promote a band, companies try to ensure that they only make records that will sell, hence the notorious difficulty faced by new musicians trying to break in to the business. Again, contrast this with the ease with which anybody's songs can be placed on the Internet for downloading.

The enormous cost of making even a 'low budget' film means that access to the medium of film is even more tightly restricted than the other forms we have considered. Among the established media, only television can begin to provide any sort of broad access. The relatively low cost of video equipment compared to film, and the absence of any specialized processing requirements, means that video can be produced on a very restricted budget — although not to broadcast quality. Transmission, however, is controlled, as much as in any other medium, if not more, and opportunities for public access to broadcasting are limited to a few derisory slots on a minority channel in the UK, or one cable channel among fifty or

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 37
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

so in New York, for example. We will see in Chapters 10 and 15 that emerging network and digital video technologies mean that anyone with a video camera, a contemporary desktop computer and a Web site can broadcast their own video over the Internet.

In discussing access to multimedia production, we have concentrated on the World Wide Web. This is because methods of off-line delivery do not offer the same possibilities. Since a CD-ROM is a physical object, it must be manufactured and distributed. Additionally, a significant amount of content is needed to fill a CD-ROM. Production and distribution of CD-ROMs therefore more closely resembles that of audio CDs or videotapes than the unique forms of online delivery. Nevertheless, the relatively cheap cost of CD writers and the very low cost of the CD media itself, has meant that to some extent a new form of self-publishing has opened up for those people with sufficient funds and motivation. It is now possible for artists of all kinds to distribute high quality portfolios or catalogues of their work themselves, or for musicians to make their own cheap CDs.

While the description we have just given is incomplete and simplified, it should convey the fact that access to traditional media is highly exclusive in comparison with access to the World Wide Web — which has the bonus of supporting interactive multimedia. It is hard to imagine a publisher issuing a book directed only at pedestrians, which lists gardens that are open to the public and can be visited on foot or by public transport, but one man is preparing a charming Web site on that subject as a personal Millennium project. Unfortunately for proponents of democratic access to multimedia, putting up a Web site is not the end of the story. People also have to visit it.

Control of Multimedia

In theory, all Web sites have equal claim on our attention, but, in practice, some exert a stronger appeal than others. Recent evidence suggests that the metaphor of 'Web surfing' — following links at hazard in the hope of coming across something useful or interesting — is no longer an accurate way of describing people's behaviour (if it ever was). Most people have a limited number of sites that they visit habitually, and only a small number of sites, compared with the vast number that exist, attract more than a few visitors. Many of the most popular sites on the World Wide Web are the home

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 38
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

sites of Internet Service Providers, which typically provide a news service, little articles about technology or lifestyles, horoscopes, advertisements, and, increasingly, online shopping. In fact, these Web portals, as they are called, are online Sunday supplements, echoing the early development of film, with its impersonation of newspapers. This Sunday supplement model of a Web site is now also provided by sites such as AltaVista and Excite, which originally functioned purely as search engines.[6] The main competition to Web portals are online stores, that is, digital mail-order catalogues. They may be more convenient for home shoppers, but the form of the online catalogues closely mimics their off-line originals, and very little extra functionality is offered beyond basic search facilities.

6
We will describe search engines and Internet portals in Chapter 9.

In addition to importing established forms, the World Wide Web is also importing content from established media. In 1999, Apple launched QuickTime TV, providing streamed video over the Internet. Like a conventional cable or satellite service, QuickTime TV is organized as a collection of channels. The current collection of channels includes ABC News, Fox Sports, Warner Brothers, VH-1, Rolling Stone, WGBH, and the BBC World Service. There is little evidence of public access provision here.

It is hardly surprising that companies should follow established advertising and marketing patterns when confronted with a new medium, or that Web content providers should turn to existing conventional sources for news and entertainment. The notion that the comparative ease with which anybody can put up a Web site provides an opportunity to wrest control of the means of distribution of ideas away from the old establishment does not bear much scrutiny, and is not supported by the evidence. The pattern of domination by identifiable large companies, corporations and institutions has, by and large, been transferred intact to the Internet. However, in addition to this predictable transference to the new media, the Internet has enabled the beginnings of something more interesting: the opportunity to create new kinds of relationship between consumers and producers of multimedia. As well as the one-to-many relationship of conventional broadcasting, publication and marketing, which the Web portals echo, we are seeing a new development of few-to-few relationships, between small groups of individuals with specialized interests, who in reality may be physically very isolated from one another. The structure of the Internet and the low production cost of Web pages provide a support for such relationships which is denied by the politics and economics of the established media.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 39
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

The Control of Content

"Then we must speak to our poets and compel them to impress upon their poems only the image of the good, or not to make poetry in our city. [...] He that cannot obey must not be allowed to ply his trade in our city. For we would not have our Guardians reared among images of evil, and [...] gather many impressions from all that surrounds them, taking them all in until at last a great mass of evil gathers in their inmost souls, and they know it not."

Plato, *The Republic*, Book III

Some issues do not go away or become resolved in a hurry: more than 2300 years have passed since Plato wrote *The Republic*, and yet very similar concerns about the protection of impressionable young people are frequently expressed today. It is precisely the enduring nature of ethical problems — which deal with fundamental issues of how human beings can and should conduct themselves — and the great difficulty in finding satisfactory and practical solutions, that has fuelled the continuation of both academic and real-world debates on these subjects.

Human history to date has shown that any sufficiently complex society has sought to exert some control over what people may read, see, hear, or do, whether this is done by explicit policing or by economic or other, less tangible, means. In the twentieth century this control has usually been justified as a benevolent protective measure, and may well be broadly supported by the electorate. It is interesting that the same type of justification has been used by political systems which were radically opposed to one another — each supposedly protecting their own population from the potentially harmful influences of the other, for example in the idealogical combat known as the Cold War.

The rapid growth of the Internet, which offers an unprecedented dissemination of possibly unacceptable or unsuitable material, has given a new impetus to debates about censorship. However, the ethical issues surrounding censorship are complicated, and a very long history of discussion has produced no enduring conclusion or consensus about them. The novelty of the contemporary situation lies in the international reach of the Internet, and its availablility within people's homes. Multimedia has not yet managed to raise any new issues of content — and it seems improbable that people will have anything to express in this medium which has not already been expressed many times before in other forms. The new dimension

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 40
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

is entirely one of access and distribution. Because the World Wide Web is currently the most effective means of distributing multimedia, anybody who is involved in multimedia production may find themselves drawn into the censorship debate, even though they may be poorly equipped to resolve the difficult ethical problems which have evaded a solution for centuries, and which have only been made more complicated by the very wide reach of the Internet. If you have a serious interest in this debate you will need to do a substantial amount of further reading in ethics generally, and censorship issues in particular, to develop a well-rounded and informed view of the situation. If you prefer to concentrate your efforts elsewhere — as most people working in traditional media have chosen to do — you would be prudent to avoid being drawn in unnecessarily, but you will need to be aware of how the debate is focussed, and of the nature of the main issues.

Within a single modern state, mechanisms generally exist for controlling publication, exhibition, performance and so on, which are consistent with the political conditions of that state. The plurality of political and social systems that exist in the world means that it is unrealistic to expect a single model of censorship — whether that means no censorship, rigid centralized control, self-regulation, or any other set of mechanisms — to be acceptable everywhere. And yet, Internet content is available everywhere[7] and the way the network is organized makes it difficult to assign responsibility for the dissemination of content, even in cases where it is desired to do so.

7 Subject to the limits we described earlier.

A society's history, its state of development, political system, and religious institutions and beliefs are among the factors that determine precisely what sorts of media or multimedia content are considered objectionable within that society. Some groups of societies may have a broadly similar history and culture, deriving from certain shared roots, which will mean that their basic concerns are concentrated within the same areas. But even in these cases there is often disagreement about the precise limits of acceptability. For example, the level of female semi-nudity that has been commonplace on beaches in the south of France for many years has been, during the same period, considered indecent on the other side of the English Channel (where it is still daring, at the least), while at the same time complete nudity of both sexes is acceptable at bathing places in Scandinavia. Where there is greater

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 41
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

cultural diversity, there is a corresponding diversity of opinion about decency.

In any society there is often a substantial separation between what is acceptable in a portrayal of life — i.e. in media — and what is acceptable in reality. In some societies, much greater leniency is extended towards representations of behaviour than to the real thing, but in others there is a much closer correspondence of the restrictions which govern media on the one hand and actual human behaviour on the other. In some there are very stringent restraints on specific media, for example on image making. These different sets of standards will be further sub-divided along lines of content: a society may sanction representations of violence while punishing those who actually do violence on others, and at the same time censor images of the sexual activity which in practice that society condones. A quick look at the films of the United States in the 1930s will bear this out. At this period even happily married couples had to be shown as sleeping in separate beds, and never apparently enjoyed any intimacy beyond a kiss and a cuddle, but at the same time gangster films and thrillers portrayed scenes of violence and murder which were illegal and generally condemned in reality. When the complexity of the standards for censorship within a single society is multiplied by the number of societies within the world, and then by the cultural and religious groups within those societies, the result is a set of principles, beliefs and practices which are very diverse indeed.

There is no alternative to accepting this diversity, so how can it be accommodated in the context of the Internet and other modern communication systems? To simply permit everything is not accommodation — restrictions, even when we may not personally approve of them, are part of the diversity. Just as it would be unacceptable to impose one society's set of restrictions on everybody else, so is it unacceptable to impose one society's freedoms on everybody else. (We are not talking here about matters which fall within the scope of 'Human Rights', which is a separate area of ethics in which an international consensus has now more or less been established through, for example, the United Nations Declaration of Human Rights.)

The *Platform for Internet Content Selection (PICS)* is an attempt to provide a mechanism that supports a diversity of attitudes towards content and censorship. The difference between PICS and conventional mechanisms, such as the banning of books or the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 42
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

seizure of video tapes and magazines, is that it restricts reception, not distribution.

⇨ An established pattern for the censorship of films, as practised in the United Kingdom and other countries with similar systems, tries to combine restrictions upon reception with some control of distribution. In extreme cases censors can refuse certification, effectively preventing the showing of that film in most venues, or they may require scenes to be edited out before a certificate can be awarded. These measures affect distribution, but by labelling each film which 'passes' with a certificate that defines what age group may legally be admitted to view that film, the censors also attempt to control reception — although in practice in the UK, for instance, this is not put into effect very rigorously. A significant flaw in this system, which has been addressed by certain video tape distributors, is that no indication is given of the nature of the content which was deemed unsuitable. This makes it very difficult for anyone to exercise their own control over what they are exposed to at the movies, unless they have access to information about the film's level of violence, or whatever, from other sources.

Acknowledging the virtual impossibility of preventing material that is unacceptable to somebody somewhere appearing on the World Wide Web, PICS aims to allow the user themselves to prevent unacceptable material reaching them. The underlying idea is simple: labels are associated with each Web page, to provide a rating of its content that can be used to assess its desirability. Undesirable material is rejected by screening software that is either incorporated in a Web browser, or implemented at a lower level to intervene in all network traffic carrying Web pages. All that PICS stipulates is a standard format for content labelling; it says nothing about what the labels should contain, nor how software should deal with them. PICS thus allows for arbitrary criteria to be used for assessing content, and arbitrary strategies for dealing with it.

PICS may sound like a laudably flexible, value-independent, means of blocking the reception of certain sorts of material, but it achieves this by deferring the difficult decisions. Who does determine the criteria for blocking, who ensures that software can apply the criteria, and who attaches the necessary labels to Web pages? In theory, PICS would allow pages to be labelled with, say, a green rating, so that pages that advocated environmentally damaging practices and policies could be filtered out. For this to work, filtering software would have to be made aware of the existence of the green label, so that it could be configured to use it as a filtering criterion.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 43
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

A PICS *service description* is a document that specifies the format of some rating service's labels. Filtering software can download service descriptions so that they can allow users to specify controls based on the labels they describe.

Most filtering is performed by Web browsers which, by default, only understand the labelling system of the Recreational Software Advisory Council (RSAC). This system is restricted to a crude set of measures of violence, sex, nudity, and language. Other rating systems can be used if appropriate service descriptions are available, but there is no guarantee that any sites will be labelled in accordance with them, whereas RSAC ratings are widely used. In practice, therefore, instead of the flexible, multi-dimensional content screening that PICS promises, we find the crude and culturally specific RSAC system.

The PICS architecture allows labels to be attached either by the page's author, or by some third party vetting organization. Where third parties do the labelling, a mechanism is available for associating labels with URLs without attaching them to the page itself, and for enabling the label to be inspected, to determine whether to allow access to the page. The green label could only be applied if concern about green issues was so widespread that Web authors felt an obligation to include green ratings, or if an ecological organization was able to set up a ratings bureau and vet Web pages. (There are more Web pages than people on this planet.) However, if there is no third party who shares your concerns, and you do not necessarily trust Web authors to label their pages, you have no means of preventing material you do not want to see from reaching you, except by complete avoidance of anything you think may be unacceptable. Even if a third party service description does embody some criteria you wish to apply to content, you have to trust the labelling service. There is no regulation of such services, so there is room for abuse. There have been allegations that some labelling bureaux have blocked access to pages critical of their own policies, for example.

PICS labels could be used for a number of purposes other than those for which they were intended. For example, it would be trivial to write a Web searching program which sought out Web pages with certain types of content, simply by looking at the PICS labels. Ironically, the mechanism intended to restrict access to certain material is itself the means by which that material may be most easily located.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 44
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

A New Medium

During the very brief history of digital multimedia to date, designers have consistently looked back to older forms for inspiration, adapting established ideas to the digital realm. This has its good side — the fundamental principles of fine typography, or the conventions of film editing provide a firm foundation for their digital counterparts — but it means that, as yet, new purely digital forms have hardly begun to emerge.

Nowhere is this more evident than on the World Wide Web. When the first browsers became available for personal computers, all Web pages looked the same. Dominated by text in Times Roman, with rudimentary layout and a few images, these pages were distinguished from printed papers only by the presence of links to other pages that looked just the same. Information content was all. Now, several years later, the Web can support elaborate layout, typography, animation, video, sound, and interactivity. Yet behind the different sorts of bells and whistles that these new elements permit, there is an essential uniformity to Web sites. They are electronic brochures, shopping catalogues, and Sunday supplements, whose multimedia elements are seen as embellishments whose primary purpose is to attract attention and gain hits. During the heyday of the multimedia CD-ROM, the same kind of formal limitation occurred: it was an electronic presentation, reference book, or again, a brochure.

These limitations have come to define a Web site; the boundaries of the form have become the boundaries of the concept. The technology we will describe in the remainder of this book is not contained by those boundaries.

Consider an Internet application that is far removed from an electronic brochure, the SETI@home project, which is based at the University of California at Berkeley. SETI stands for the Search for ExtraTerrestrial Intelligence. One way of searching for extraterrestrial intelligence is to look for radio signals concentrated in a narrow wavelength band. Orbiting radio telescopes that continually scan the sky generate enormous quantities of data about the radio signals that are reaching the Earth. Processing their raw data to search for signals with characteristics that might indicate an intelligent origin requires correspondingly enormous amounts of computer power. Searchers for extraterrestrial intelligence are

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 45
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

not generously funded, so research groups cannot command the necessary amount of time on supercomputers. But nowadays, supercomputers are not needed, if a sufficient number of less powerful computers can be made to cooperate on the computation.

The Internet connects a huge number of computers, most of which are not doing anything useful most of the time. Quite a lot of the time, many computers are displaying flying toasters, goldfish, and space-filling curves, allegedly to prevent their monitor phospors burning out during periods of inactivity. The SETI@home project makes use of that time to look for intelligent signals from outer space. Instead of running a conventional screen saver, participants are running special software that downloads a small portion of the radio telescope data and, at times when the screen saver would normally kick in (i.e. when the user has gone to a meeting, or is having a long phone call), the program does some data analysis.[8] When it has finished with its current data set, it sends the results of the analysis back to the project's headquarters, and gets some more.

8
It shows a graphic display of its progress, so you still get your screen saved.

Here is a project that exposes the true nature of the Internet: it is a collection of connected *computers*. This is what the restricted concepts of 'Web site', and 'multimedia CD-ROM' omit. Naturally, they depend on the processing power of computers to display media and handle interactivity, but in doing so, they reduce the computer to a flexible media player, and deny its power to perform computation. To develop multimedia forms that accommodate and exploit that power will require a leap of imagination and understanding, comparable to the leap of imagination that took computer graphics and placed it at the centre of the user interface. Until that happens, the promise of digital multimedia is still unfulfilled.

Further Information

Some of the issues raised in this chapter are taken up from a different perspective by Brent MacGregor in his Afterword.

If you are interested in the cultural theory of multimedia, [Bar77] is a good starting point. For analyses of established media, consult [Wil74] (televison) or [Kaw92] (film); [McC96] considers the impact of new technology on craftsmanship, and [Lis95] looks at digital

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 46
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

images. [RM96] is a short introduction to the mechanics of PICS. [EF99] and [App93a] both provide a management-oriented view of multimedia that complements our technical treatment. [Fau98] gives a concise introduction to interface design.

Exercises

1. Is the idea of making a film of this book an absurd one?

2. The late Kingsley Amis reportedly once asserted that he would never read another book unless it began *A shot rang out!*. How would this opening be realized in

   (a) A radio play;

   (b) A stage play;

   (c) A film;

   (d) An animation;

   (e) A single created image;

   (f) A photograph;

   (g) A comic strip;

   (h) A song;

   (i) A symphony?

   If you have access to suitable facilities, produce a rough version of each of your answers.

3. A commonly used means of multimedia delivery is the *information kiosk*, a computer system with limited functionality provided for the public to learn about, for example, the exhibits in a museum, or tourist facilities in a city. Discuss whether information kiosks are a qualitatively different form of delivery from those described in this chapter.

4. Choose a Web site and a multimedia CD-ROM available to you. For each of them, identify at least five (and preferably ten) ways in which you think it could be improved. State the criteria you use to judge them by.

5. Look closely at the user interface of any application of your choice, and identify ten features of that interface which you

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 47
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

personally do not like. Explain the reasons for your choices, and suggest possible improvements.

6. Choose a straightforward subject, such as a guide to guest accommodation in your neighbourhood. Design a simple one-page layout of graphical material and text, which optimises intelligibility and access to information for potential users of all levels of ability.

7. We stated on page 13 that 'ultimate control over "the content and flow of information" remains with the producer not the user'. Ignoring feasibility, discuss whether it would be desirable for true control to be transferred to the user.

8. If you are working within a group of people, have each member of the group devise their own set of six filtering labels, which they would choose to apply to a Web site of their making for the guidance of users, and then compare and discuss results. If you are in a position to do so, extend this exercise to include other groups in other countries.

9. A committed vegetarian couple wish to prevent their children from being exposed to images of meat on the World Wide Web. What difficulties would they face in using PICS ratings for this purpose?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 48
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

2 Enabling Technologies

The production and consumption of digital multimedia as we know it depends on the ability of digital computers to perform operations at high-speed. In order to benefit from this ability, media data must be in digital form. That is, the rich variety of sensory inputs that make up images, text, moving pictures, and sound, must be reduced to patterns of binary digits inside a computer. Once this remarkable transformation has been effected, programs can be used to change, combine, store and display media of all types. Furthermore, the same data can be transmitted over networks, to be distributed anywhere in the world, or conveyed to remote destinations on removable storage such as CD-ROMs or DVDs.

Although a digital representation of data is fundamental, general purpose computers are not the only devices that can manipulate digital data. Digital video can be played from DVD by special players, digital TV only requires a simple set-top box, and digital audio can be played from CD by any CD player. At present, only a programmable computer can add full interactivity to multimedia, but it seems almost inevitable that, as the technology matures, consumer devices that function as multimedia players will be

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 49
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

brought to market. It is to be expected that such devices will be significantly cheaper than personal computers, and that they may bring about the transformation of digital multimedia into a true form of mass communication.

Digital Representations

Computers are built out of devices that can only be in one of two states. Physically, this means that certain points in the devices' circuits are only stable at one of two well-defined voltages. In more abstract terms, we can say that these devices store and operate on *bits*, units of data that can only have one of two values. We might call these values 0V and 3.5V, or on and off, true and false, yin and yang, but conventionally we denote them by 0 and 1. Bits are usually grouped into larger units such as *bytes*, which consist of an ordered sequence of eight bits, or *words*, whose length depends on the particular model of computer, but is often four bytes, that is, thirty two bits. Adopting the convention that a bit is either 0 or 1 suggests an interpretation of these larger groups: they can be read as numbers to base 2, whose digits are the component bits. Thus, the byte containing the eight bits 0, 1, 1, 0, 0, 0, 0, and 1 can be read as the binary number 01100001, which is 97 in decimal. Not only can we interpret bytes and words in this way, we can also build electronic devices that perform the basic arithmetic operations of addition, subtraction, and (with a bit more effort), multiplication and division, and produce an answer in the same format. Hence the once popular notion that computers are just giant high-speed calculating machines.

There is, however, nothing intrinsically numerical about bits and bytes. It is only the way that we choose to interpret them, and the operations that we build into our computers, that make these bits and bytes into numbers. We can choose to interpret patterns of bits in different ways, and this is how the data belonging to different media can be represented digitally. The pattern 01100001 might denote a particular shade of grey occurring in an image, if we wrote software and built display hardware to interpret it in that way.

It is easiest to see what is going on if we describe the interpretation of bit patterns in terms of the interpretation of numbers, since we know how numbers and bit patterns can be associated, and numbers

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 50
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

are easier to write down, and have familiar properties. Thus, we represent characters of text by associating a unique number with each letter, digit or other sign we need, in the manner of a code. The widely used ASCII character set, for example, is an association between characters and numbers; it assigns the value 97 (i.e. the bit pattern 01100001) to the lower case letter a, 98 to b, and so on, for 96 different printable characters. For this association to mean anything, we must arrange that hardware — such as keyboards and printers — and software behave in a way that is consistent with it. For example, when you press the key labelled A on your keyboard without holding down the shift key, the number 97 (or rather a sequence of electrical pulses that can be interpreted as the corresponding pattern of bits) is sent to your computer.

As we will see in later chapters, a similar association can be made between numbers and quantities such as the brightness of an image at a point, or the instantaneous amplitude of a sound wave. Although the association is essentially arbitrary, there will normally be some structure to the way numbers are assigned to the quantities they represent: e.g. numbers denoting high brightness have greater values than those denoting low brightness. Such properties tend to fall out naturally from the nature of the quantities represented.

⇨ Ada Augusta, Countess of Lovelace, appears to have understood the nature of digital representations in 1844, when she wrote:

> "[Babbage's Analytical Engine] can arrange and combine its numerical quantities exactly as if they were letters or any other general symbols; in fact it might bring out its result in algebraical notation *were provisions made accordingly.*" [Our italics.]

Bits are arranged into bytes, and in the memory of a computer, bytes are arranged in a linear sequence so that each byte[1] can be identified by its position in the sequence, which we usually call its address. Addresses behave like numbers, so they can be represented by bit patterns and stored and manipulated like other quantities. This makes it possible to organize collections of bytes into *data structures.* For example, a black and white image is often represented by the values corresponding to the brightness at each point on a fine rectangular grid. We can store these values in a sequence of bytes, and then use the address of the first byte to access the image data; a simple computation allows us to work out the address of the byte corresponding to any grid point, and

1 Some computers use a different addressable unit, but the principle is the same.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 51
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

access the stored value. If we need a sequence of such images, representing successive frames in an animation, say, we can store with each image the address of the next and previous ones, so that we can easily traverse the sequence in either direction.

The most important interpretation of bit patterns in a computer is only incidentally related to multimedia: bit patterns can represent instructions that cause the processor to carry out operations on values stored in memory. Again, this interpretation relies on the fact that the hardware is constructed so that the intended effect is achieved when the instruction is presented to the processor. Because instructions are bit patterns, sequences of instructions — *programs* — can be stored in memory and executed. This is the defining characteristic of a computer: it is a *stored program* machine. This is what makes it possible for the same machine to be used for many diverse tasks, from the computation of tax allowances to the precise editing of video footage.

To reiterate a point we made in Chapter 1: the common representation as collections of bits means that data belonging to all media can be manipulated by programs, both individually and in combination. At present, there are established ways of representing text, images, sound, video and animation in bits; we can be fully confident that any media developed in the future will also be representable digitally.

Digitization

Multimedia data sometimes originates in digital form, as it does when images are made using a painting program, but it is frequently necessary to convert from some analogue representation to a digital one, for example, by scanning an image.

A banal illustration of the differences between analogue and digital representations is provided by contemporary time-pieces. Figure 2.1 shows the face of an analogue wristwatch and the display of a digital alarm clock. The minute hand of the watch moves steadily, and can point anywhere on the rim; it is not constrained to point only at the marks indicating exact minutes. This alarm clock can only display whole numbers of minutes, and its displayed value only changes once every sixty seconds. The digital representation is more convenient than the analogue one for precise computation: for example, if we want to know how long it is until half past twelve,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 52
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046



Figure 2.1
Analogue and digital representations

we need only subtract the displayed time from 12:30, whereas for the analogue watch we must estimate the difference based on the angular difference between the current position of the hour hand, and a position half way between twelve and one (or else read the watch, and convert the time to its digital form in our heads). This particular analogue representation is more detailed: the alarm clock cannot show how far between 7:47 and 7:48 it is, whereas this can be estimated (to an arbitrary level of precision) from a close examination of the position of the watch's minute hand.

⇨ If you are feeling contrary, you will point out that the watch's hands do not really move continuously — there are gears and stepping motors and things inside the watch that make it actually move in a series of small jerks. (Nowadays, new watches are almost all digital inside, and so, even where the display is analogue, the hands can only move in discrete steps, as is evident if you watch the second hand. Even older, spring-powered, mechanisms do not move truly continuously, though, because of the finite size of the moving parts.) There is no such thing as continuous physical movement, if only because of the quantum nature of electrons and photons; the infinitesimally small is a mathematical abstraction. However, as in the case of the watch, a continuous (analogue) model of behaviour is often more tractable than a digital one, and provides a more accurate description of what is going on than any manageable discrete (digital) model could do.

In multimedia, we encounter values that change continuously in several ways. For example, the amplitude of a sound wave varies continuously over time, as does the amplitude of an electrical signal produced by a microphone in response to a sound wave. The brightness of any point on a black and white photograph could in principle have any value, though in practice this will be restricted by the physical constraints of the process of photography. As you see, we may be measuring different quantities, and they may be varying either over time or over space (or perhaps, as in the case of moving pictures, both). For this general discussion, we will follow tradition, and refer to the value we are measuring, whatever it may be, as a *signal*, not usually distinguishing between time-varying and space-varying signals.

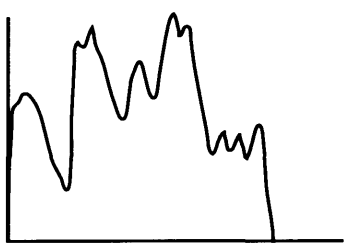


Figure 2.2
An analogue signal

You will observe that, when we have a continuously varying signal, such as the one shown in Figure 2.2, both the value we measure, and the intervals at which we can measure it, can vary infinitesimally. In contrast, if we were to convert it to a digital signal, we would have to restrict both of these to a set of discrete values. That is, *digitization*

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 53
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

— which is what we call the process of converting a signal from analogue to digital form — consists of two steps: *sampling*, when we measure the signal's value at discrete intervals, and *quantization*, when we restrict the value to a fixed set of levels. Sampling and quantization can be carried out in either order; Figure 2.3 shows a signal being first sampled and then quantized. These processes are normally carried out by special hardware devices, generically referred to as *analogue to digital converters (ADCs)*, whose internal workings we will not examine. We will only consider the (almost invariable) case where the interval between successive samples is fixed; the number of samples in a fixed amount of time or space is known as the *sampling rate*. Similarly, we will generally assume that the levels to which a signal is quantized — the *quantization levels* — are equally spaced, although, in this case, other arrangements are sometimes used.

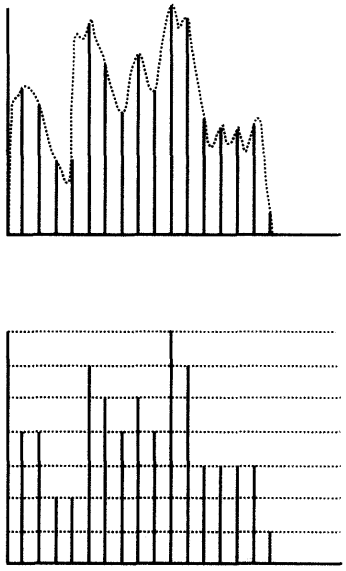


Figure 2.3
Sampling and quantization

One of the great advantages that digital representations have over analogue ones stems from the fact that only certain signal values — those at the quantization levels — are valid. If a signal is transmitted over a wire or stored on some physical medium such as magnetic tape, inevitably some random noise is introduced, either because of interference from stray magnetic fields, or simply because of the unavoidable fluctuations in thermal energy of the transmission medium. This noise will cause the signal value to be changed. If the signal is an analogue one, these changes will be more or less undetectable — *any* analogue value is legitimate, and so a signal polluted by noise cannot be distinguished from a clean one. If the signal is a digital one, though, any minor fluctuations caused by noise will usually transform a legal value into an illegal one, between quantization levels. It is then a simple matter to restore the original signal by quantizing again. Only if the noise is sufficiently bad to alter the signal to a different level will an error in the transmission occur. Even then, because digital signals can be described numerically, schemes to detect and correct errors on the basis of arithmetic properties of groups of bits can be devised and implemented. Digital signals are therefore much more robust than analogue ones, and do not suffer degradation when they are copied, or transmitted over noisy media.

However, looking at Figure 2.3, it is evident that some information has been lost during the digitization process. How can we claim that the digitized result is in any sense an accurate representation of the original analogue signal? The only meaningful measure of

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 54
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

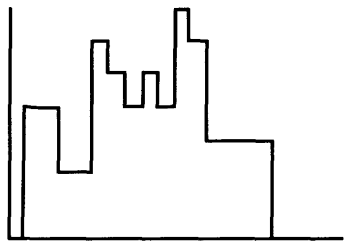


Figure 2.4
Sample and hold reconstruction

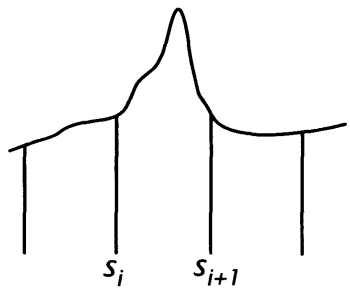


Figure 2.5
Under-sampling

accuracy must be how closely the original can be reconstructed. In order to reconstruct an analogue signal from a set of samples, what we need to do, informally speaking, is decide what to put in the gaps between the samples. We can describe the reconstruction process precisely in mathematical terms, and that description provides an exact specification of the theoretically best way to generate the required signal. In practice, simpler methods that can easily be implemented in fast hardware are used.

One possibility is to 'sample and hold': that is, the value of a sample is used for the entire extent between it and the following sample. As Figure 2.4 shows, this produces a signal with abrupt transitions, which is not really a very good approximation to the original. However, when such a signal is passed to an output device, such as a CRT display or a loudspeaker, for display or playback, the lags and imperfections inherent in the physical device will cause these discontinuities to be smoothed out, and the result actually approximates the theoretical optimum quite well. (However, in the future, improvements in the technology for the playback of sound and picture will demand matching improvements in the signal.)

Clearly, though, if the original samples were too far apart, *any* reconstruction is going to be inadequate, because there may be details in the analogue signal that, as it were, slipped between samples. Figure 2.5 shows an example: the values of the consecutive samples taken at s_i and s_{i+1} are identical, and there cannot possibly be any way of inferring from them the presence of the spike in between those two points — the signal could as easily have dipped down or stayed at the same level. The effects of such *under-sampling* on the way in which the reconstructed signal will be perceived depend on what the signal represents — sound, image, and so on — and whether it is time-varying or space-varying. We will describe specific instances later. Suffice it to say, for now, that they are manifested as distortions and artefacts which are always undesirable.

It is easy enough to see that if the sampling rate is too low some detail will be lost in the sampling. It is less easy to see whether there is ever any rate at which we can be sure samples are close enough together to allow the signal to be accurately reconstructed, and if there is, how close is close enough. To get a better understanding of these matters, we need to consider an alternative way of representing a signal. Later, this will also help us to understand some related aspects of sound and image processing.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 55
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

You are probably familiar with the idea that a musical note played on an instrument consists of waveforms of several different frequencies added together. There is the fundamental, which is the pitch associated with the note, but depending on the instrument, different numbers of harmonics, or overtones, are also present, and these give the note its distinctive timbre. The fundamental and each harmonic are pure tones — sine waves of a single frequency. Any periodic waveform can be decomposed into a collection of different frequency components, each a pure sine wave, in a similar way; in fact, with a little mathematical gymnastics, assuming you don't mind infinite frequencies, any waveform can be decomposed into its frequency components. We are using the word 'frequency' in a generalized sense, to go with our generalized concept of a signal. Normally, we think of frequency only in the context of time-varying signals, such as sound, radio, or light waves, when it is the number of times a periodic signal repeats during a unit of time. When we consider signals that vary periodically in space, such as the one shown in Figure 2.6, it makes equal sense to consider its frequency as the number of times it repeats over a unit of distance, and we shall often do so. Hence, in general discussions, frequencies, like signals, may be either temporal or spatial.[2]

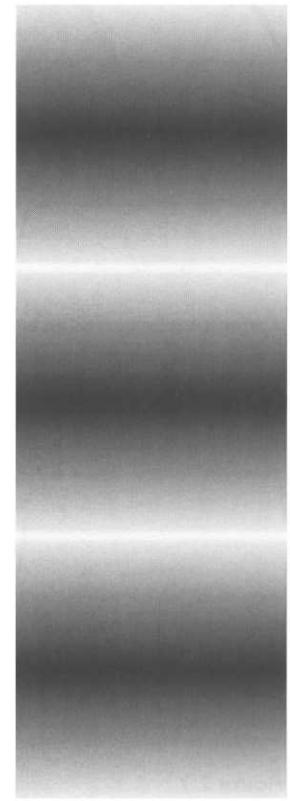


Figure 2.6
A periodic fluctuation of brightness

Figure 2.7 shows one of the classic examples of how pure sine waves at different frequencies combine to produce more complex waveforms. Starting with a pure sine wave of frequency f, we successively add components to it with frequencies of $3f$, $5f$, $7f$, and so on, whose amplitudes are one third, one fifth, one seventh,... the amplitude of the original signal. As you can see, as we add more 'harmonics', the signal begins to look more and more like a square wave; the more frequency components we add, the better the approximation.

Although it may not have the same immediate perspicuity, we could use the frequencies and amplitudes of its components to represent our signal. The collection of frequencies and amplitudes is the signal's representation in the *frequency domain*.[3] It can be computed using a mathematical operation known as the *Fourier Transform*. The result of applying the Fourier Transform to a signal can be displayed, like the original signal itself, in the form of a graph, where the horizontal axis represents frequencies and the vertical axis amplitude. A typical signal's frequency spectrum, as this representation is called, will consist of a set of spikes at different frequencies, corresponding to the different components.

2
You will probably realize that a spatially varying signal may vary in two dimensions, not just one. We will consider this complication in Chapter 5, but it can safely be ignored for now.

3
Strictly speaking, we also need to know the phase of the components, since it is sometimes necessary to displace them relative to each other, but we will ignore this complication, too.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 56
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

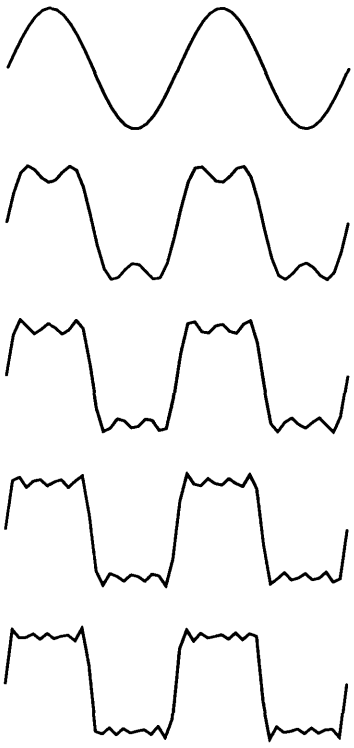


Figure 2.7
Frequency components of a square wave

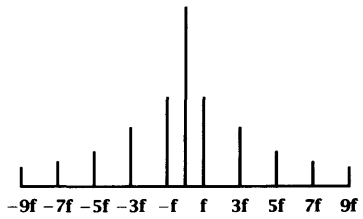


Figure 2.8
Square wave transformed into the frequency domain

Figure 2.8 shows our square wave in the frequency domain. You will notice that it includes negative frequencies. There is no need to worry about this: negative frequencies are a notational convenience that allow us to deal with phase shifts; where none is present, as here, the representation will be symmetrical, with the negative frequencies matching positive ones. There is also a spike at a frequency of zero, which is called the *DC component*, from the long-standing use of frequency domain representations in electrical engineering. You can think of it as the average value of the signal.

The square wave example demonstrates a phenomenon that can be shown to be generally true: *higher frequency components are associated with abrupt transitions.* As we add higher frequencies, the leading and falling edges of the waveform become more nearly vertical. Looking at this from the other end, as we omit high frequency components, such abrupt changes get smoothed out. Hence, operations such as sharpening or smoothing an image can be described and implemented in terms of *filters* that remove certain frequencies — this is of fundamental importance in the processing of graphics, as we will see in Chapter 5.

⇨ The *Inverse Fourier Transform*, as its name suggests, performs the opposite operation to the Fourier Transform, taking a signal from the frequency domain to the time domain.

Returning to the question of whether there is a sampling rate that guarantees accurate reconstruction of a signal, we can now give a precise answer. The *Sampling Theorem* states that, if the highest frequency component of a signal is at f_h, the signal can be properly reconstructed if it has been sampled at a frequency greater than $2f_h$. This limiting value is known as the *Nyquist rate.*

⇨ Some authors, especially in the field of audio, use 'Nyquist rate' to denote the highest frequency component that can be accurately reproduced. That is, if a signal is sampled at f_s, their Nyquist rate is $f_s/2$. The fact that the term is used with both meanings is unfortunate, but any ambiguity is usually easily resolved by context. We will always use the term in the sense defined in the main text.

The proof of the Sampling Theorem is technical, but the essence of the underlying effect can be illustrated simply. Suppose we have a circular disk, with a single radial line marked on it, which is spinning in a clockwise direction at a rate of n rotations per second, and suppose that we 'sample' this rotation as a movie camera would,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 57
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

by taking snapshots of the disk at equal time intervals. Figure 2.9 shows the snapshots we would obtain by sampling $4n$ times a second. Looking at this sequence (ignoring, for now, the dashed boxes) it is clear that the disk is rotating clockwise, and you can imagine that if these images were successive frames of a film, we would see the disk rotating in its proper direction when the film was projected. Considered as a periodic signal, the rotating disk has a frequency of n and we have sampled at $4n$, comfortably above the Nyquist rate. Consider now what happens if we sample at a rate of $4n/3$; the samples we obtain are the ones identified by dashed boxes. Looking at the sequence comprising only those samples, it appears from the successive positions of the line more as if the disk is rotating anti-clockwise at a rate of $n/3$. (The phenomenon may be familiar to you from Western movies, in which stagecoach wheels frequently appear to rotate backwards, because the rate at which the film frames were shot is less than the Nyquist rate relative to the actual rotational speed of the wheels.) Note that we must go beyond the Nyquist rate: if we sample our disk at a rate of exactly $2n$, we get samples in which the line alternates between the 12 o'clock and 6 o'clock positions, so that it is impossible to determine whether the disk is rotating clockwise or anti-clockwise.

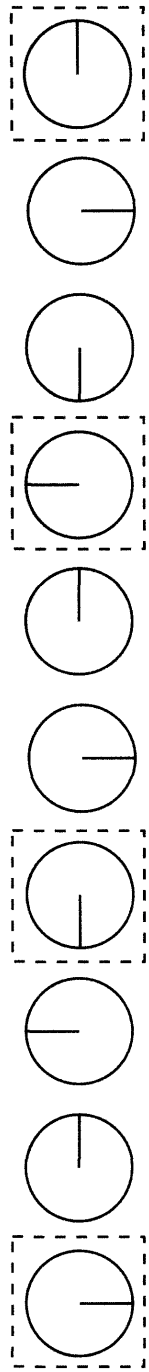


Figure 2.9
Sampling and undersampling

> The Sampling Theorem only holds true if the signal is reconstructed from its samples in a particular way — technically, it is necessary to perform the operation in the time domain in such a way that it is equivalent to multiplying the sampled signal by a perfect pulse function in the frequency domain, that is, a function whose value is 1 between a specific pair of frequency values, and 0 everywhere else. By manipulating Fourier Transforms and their inverses, it is possible to arrive at a definition of a function that must be applied to the samples in the time domain to achieve this effect. Unfortunately, this function has non-zero values at points arbitrarily far from the origin. It is therefore impossible to realize the reconstruction operation in this way. In reality, we can only approximate it, with functions that are finitely wide. The practical effect is that it is necessary to sample at a rate still higher than the Nyquist rate to achieve perfect reconstruction.

In general, if we *undersample* a signal — sample it at less than the Nyquist rate — some frequency components in the original will get transformed into other frequencies when the signal is reconstructed, just as our rotating disk's frequency was transformed when we undersampled it. This phenomenon is known as *aliasing*, and is perceived in different ways in different media. With

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 58
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

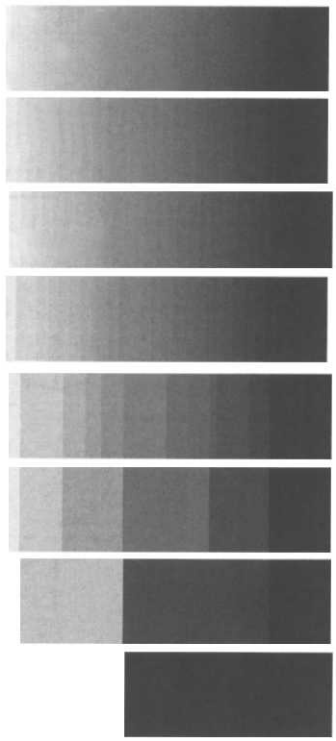


Figure 2.10
256, 128, ..., 2 grey levels

sound, it is heard as distortion; in images, it is usually seen in the form of jagged edges, or, where the image contains fine repeating details, Moiré patterns (see Plate 2); in moving pictures, temporal undersampling leads to jerkiness of motion, as well as phenomena similar to the retrograde disk just described.

The effects of an insufficient number of quantization levels are generally easier to grasp intuitively than those of inadequate sampling rates. If we can only represent a limited number of different values, we will be unable to make fine distinctions among those that fall between. In images, it is as if we were forced to make do with only a few different colours, and so had to use, say, crimson for every shade of red we needed. The difference between scarlet and carmine would be lost, and any boundaries between areas of those colours would be elided. The effect on black and white images can be seen clearly in Figure 2.10 which shows a gradient swatch using 256, 128, 64, 32, 16, 8, 4, and 2 different grey levels. The original gradient varies linearly from pure white to pure black, and you can see how, as we reduce the number of different greys, values band together as they are quantized increasingly coarsely. In a less regular image, the effect manifests itself as posterization, more formally known as brightness contouring, where coloured areas coalesce, somewhat like a cheaply printed poster.

The numbers of grey levels in our example were not chosen at random. The most common reason for limiting the number of quantization levels (in any type of signal) is to reduce the amount of memory occupied by the digitized data by restricting the number of bits used to hold each sample value. Here we have used 8, 7, 6, 5, 4, 3, and 2 bits, and finally a single bit. As you can see, although the effect is visible with 128 greys, it only becomes intrusive when the number of levels falls to 32, after which deterioration is rapid.

When sound is quantized to too few amplitude levels, the result is perceived as a form of distortion, sometimes referred to as *quantization noise*, because its worst manifestation is a coarse hiss. It also leads to the loss of quiet passages, and a general fuzziness in sound (rather like a mobile phone in an area of low signal strength). Quantization noise is clearly discernible when sound is sampled to 8 bits (256 levels), but not (except in the opinion of some audiophiles) at the 16 bits (65536 levels) used for audio CDs.

⇨ If you have ever used a drum machine or sequencer for making music, you will have met another form of quantization: these devices quantize notes in time, so that they fall on exact beats. A

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 59
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

poor drummer can be brought into a tight rhythm by quantizing to sixteenth notes, for example. However, if you quantize to eighth notes, any funky syncopated rhythms will be lost — there are too few quantization levels. (On the other hand, if your drummer is really egregious, and you quantize to sixteenth notes when they are playing in straight eights, you may let in some of their mistakes.)

Hardware Requirements

If representing media digitally is conceptually so simple, and if digitization is so well understood mathematically, why is it that digital multimedia is such a relatively new development in computing? What we have so far failed to address is the resource requirements. As subsequent chapters will show, sound, video and most images require large amounts of disk space and memory. Time-based media also have to be captured and played back in real time, so the very large quantities of data to be handled imply a requirement for fast processors and data buses capable of high speed transfers. Accessing media data over a network demands high bandwidth. It is only recently that fast enough computers with sufficient memory and large high-speed disks have become available. At the time of writing, in 1999, domestic network connections are still not fast enough to support multimedia at anything but the minimum acceptable quality. As new access technologies boost Internet bandwidth, a further explosion in the use of multimedia is widely anticipated.

When we look at the hardware requirements of multimedia, there are two distinct cases to consider: requirements for multimedia consumption and requirements for production. Realistically, in the case of consumption, it is more sensible to consider the capabilities of typical domestic systems, and the limitations which those impose, than some putative set of requirements for ideal playback.

A series of specifications produced by a consortium of PC manufacturers has defined the *multimedia PC (MPC)*. The latest of these defines the Level 3 MPC as having a 75MHz Pentium processor,[4] with 8 Mbytes of RAM, running Windows 3.11 (or a binary compatible OS), with at least 500Mbytes of hard disk, a 4× CD-ROM drive, CD quality audio output, and the capability to play quarter-frame video

4 Or a lesser processor, provided additional hardware support for video playback is provided.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 60
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

at full frame rate in 15-bit colour. This probably sounds like a bit of a joke, but you should be aware that at least some owners of machines satisfying the MPC specification probably believe that their hardware is capable of 'doing multimedia'.[5] More realistically, a 'revision C' iMac, running MacOS 8 on a 333MHz PowerPC 750 processor, with 32 Mbytes of RAM, 6 Gbytes hard disk, a 24× CD-ROM drive, a built-in V90 modem, built-in stereo speakers, a powerful graphics accelerator card and a multiple scan monitor capable of 24-bit colour (or an equivalent Windows 98 PC system based on a Pentium II or III processor) represents the sort of hardware available to digital multimedia consumers at the end of the 1990s. Even these relatively powerful machines cannot play back full-frame, full-motion video at VHS quality without additional hardware.

5
In fact, the label 'multimedia PC' can be attached to a machine satisfying the Level 2 or even Level 1 specifications — they will have a CD-ROM drive and a sound card.

While one can be confident that the best available desktop consumer machines will soon be capable of flawless video playback, at the same time a new class of less powerful devices is coming into use. Hand-held 'personal digital assistants' (PDAs) are evolving from simple digital address books into capable computational systems. Their small size makes it difficult to achieve the same performance as desktop machines, and imposes limitations on the amount of storage that can be accommodated and the size of the display. The communication facilities of these devices are often based on the cellular telephone network, which provides less bandwidth than the fixed network. It may seem foolish to even consider using PDAs for multimedia, yet it seems inevitable that they will increasingly be used for Internet access, including Web browsing. This presents a challenge to Web designers to produce pages that are accessible both to powerful desktop machines and to PDAs. At the system level, it means that multimedia architectures must be scaleable, allowing different versions of a production to be made available transparently to devices with different capabilities. We will return to this question in Chapter 15.

Although the hardware requirements for high quality multimedia production are more demanding than those for its consumption, a great deal of multimedia is produced on high-end PCs running Windows NT and on Macintosh computers. These systems are often augmented with special-purpose graphics acceleration hardware and input devices for video and audio. For example, at the time the iMac described earlier was a typical consumer machine, a G3 Power Macintosh with a 450 MHz version of the same processor fitted

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 61
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

with 1 Mbyte of high-speed cache memory, 128 or 256 Mbytes of main memory, and a built-in 128-bit graphics accelerator board with 16Mbytes of video memory, would be typical of the better desktop machines used for multimedia production. This would probably be fitted with additional high-speed disks, a video capture card, and other specialized peripherals, as described below. More modest equipment can also be used to produce multimedia — many Web pages are produced on ordinary desktop PCs — but the range of material that can be created and manipulated on such machines is limited. At the opposite extreme, high-powered workstations, such as those produced by SGI, are also used, especially for computationally intensive tasks, such as rendering 3-D animation and video effects, but their use is more common in the film and television industries. The incorporation of special-purpose array processing instructions into both Pentium and PowerPC processors (MMX and AltiVec, respectively) is blurring the distinction between high-end personal computers and workstations.

Raw processing power, high-speed data buses, large main memories, and powerful graphics boards are a necessity for producing multimedia content; fast high-capacity secondary storage is equally necessary. We have already noted that sound, video and images require large amounts of storage. During their preparation, they often require substantially more storage than the finished product will need. For example, images may be composed in layers (see Chapter 5), and 'flattened', by combining the separate layers, for delivery. Each layer occupies roughly the same amount of space as the final image, and designers often use up to a hundred layers. High quality audio and, above all, video source material can occupy a very large amount of storage indeed. The degree of compression required for final delivery, especially of video or over a network, usually entails a loss of quality that cannot be tolerated during production, so less compressed, and therefore much larger, files will be in use throughout the production process, with high compression only being applied at the final stage.

Manufacturers are regularly increasing the capacity of hard disks, and several can be used together if necessary. The speed of data transfer to and from large disks is a more serious limiting factor, particularly for digital video, where transfer rates in excess of 5 Mbytes per second — preferably well in excess, up to 30 Mbytes per second — are desirable during capture. The IDE and SCSI-1 disk drives normally fitted to desktop machines cannot cope with

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 62
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

such rates. However, newer revisions of the SCSI standard provide much higher data rates: fast and wide SCSI-2 supports 40 Mbytes per second, while Ultra SCSI-2 can support 80 Mbytes per second; development of the SCSI interface standard is continuing, with a specification of a 160 Mbytes per second version already drafted (in 1999). An alternative standard for connecting mass storage devices is *FireWire*, or IEEE 1394 as it is more formally known, which not only provides transfer rates up to 50 Mbytes per second, but is also used as the standard interface for connecting consumer electronic devices, especially DV cameras (see Chapter 10) and digital stills cameras to computers.

⇨ For video capture it is essential that the data transfer to disk be maintained at a constant rate, or frames may be lost. This is only feasible if the free areas on the disk are contiguous. If a disk becomes fragmented, so that free areas are scattered over its surface in small pieces, the data transfer will be intermittently interrupted, as the disk seeks to a new position. It is important, therefore, that disks to be used for video capture be periodically de-fragmented, using a disk optimizing utility, or simply completely erased before a clip is captured, if this is feasible.

When peripherals are connected via one of the fast SCSI or FireWire buses, the limiting factor becomes the speed at which data can be moved off or onto the disks. With current disk speeds, a single, modestly priced, high-capacity disk connected via FireWire or Fast SCSI 2 to a dedicated computer will provide adequate performance for multimedia production. If, however, it is desired to share data between workstations, using a central server and a local area network, higher performance will be required. Fibre channel or Gigabit Ethernet can provide the transfer speeds. Relatively expensive high-speed disks can be used for storage, but a popular alternative is a *RAID array*. RAID stands for Redundant Array of Inexpensive Disks, and a RAID array is indeed put together out of cheaper, relatively slow, disks: improved performance is achieved by reading and writing from and to them in parallel.

More precisely, RAID specifies eight different levels, which offer different degrees of performance and fault-tolerance. The emphasis of most of these levels is on the latter, which explains the 'Redundant' in the expansion of RAID. However, the lowest level, Level 0 RAID, specifies a 'data striping' technique, whereby a data block being written to the array is split into segments which are written to separate disks. The write operations can be overlapped,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 63
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

so that data can be written to the drive buffer faster than it can be physically transferred to any single drive in the array. Read operations can be similarly overlapped. The array looks like a single high-capacity fast drive. RAID 0 offers no protection against disk failure — if one disk fails, the whole array fails. Higher RAID levels are designed to protect against failures, by incorporating redundancy, with more or less sophistication, into the data storage scheme. Level 1 is the least sophisticated: it mirrors disks, which, instead of improving performance, decreases it. Higher levels use more sophisticated schemes, but a popular option is to combine RAID levels 0 and 1, thereby gaining the performance of data striping and the protection of disk mirroring.

⇨ Marketing literature sometimes uses the term 'AV disk', which is supposed to indicate disks that are particularly suitable for use with audio and video. Originally, this indicated that the disk did not perform periodic recalibration to compensate for thermal expansion. Such calibration operations interfere with data transfers, and since video and audio require transfer at a consistent sustained rate, disks that performed these operations would be unsuitable for use with multimedia data. Some modern disks do not perform recalibration in the same way, and the term 'AV' may be loosely used simply to indicate 'fast' — but almost all modern disks are, in themselves, fast enough anyway, except for use in servers.

In addition to large fast disks, multimedia production requires other, more specialized, peripherals. A graphics tablet with a pressure-sensitive pen is necessary for all kinds of graphical work (with the possible exception of some image correction tasks) — you cannot properly draw or paint with a mouse. A large (often 20 inch) high-resolution monitor capable of accurately reproducing millions of colours is also desirable (although it should always be remembered that the end user will probably have a smaller, and possibly lower quality, display). It is common practice to attach two monitors to a computer for preparing images, one to display the image, the other, which may be smaller, to hold the plethora of tool bars and palettes that make up the user interface of most graphics programs. High resolution scanners are commonly used for capturing images from photographs and printed media. Digital cameras are also becoming popular, since they allow photographs to be downloaded directly to disk, without the expense, time, or specialist laboratories required to process film. However, only the most expensive digital cameras offer anything like the picture

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 64
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

quality of conventional 35mm SLR cameras. A video camera and sound recording equipment are also needed for recording these media. Nowadays, this equipment may itself be digital and produce digital output that can be sent directly to the computer, over a FireWire connection, for example. Video and audio capture from conventional analogue equipment requires analogue to digital converters and compression hardware, which will be described in Chapters 10 and 12. The amount and quality of additional studio equipment — microphones, mixers, tripods and heads, lighting, and so on — required will depend on how ambitious and professional a multimedia producer is. Generally, the expectations of domestic consumers are high, so for productions intended for the domestic market, production values must be correspondingly high.

Software

We have emphasized that data of all media types are represented digitally as collections of bits, and can therefore be manipulated by programs on a computer. The things that you might naturally want to do to the bits that represent an image are not the same, though, as the things that you might naturally want to do to the bits that represent a piece of text or an audio clip. Different applications have therefore been developed for the different media types: image editing, painting and drawing programs for graphics; editors and layout programs for text; capture, editing and post-production software for video; special packages for motion graphics and animation; recording, editing, and effects programs for sound, and music synthesizers and sequencers, to name just some.

The essence of multimedia is the combination of the different media elements that can be produced by these programs. One way of carrying out this combination would be by writing a program, but, while this provides the maximum flexibility, it requires advanced programming skills (which are not always found in conjunction with the necessary skills for effective presentation of multimedia), while being repetitive and error-prone. *Authoring systems*[6] have been devised to carry out the task within an organizational framework that permits the automation of much of the work that would otherwise require programming. Authoring systems may be based on a layout model with a markup language, or on a timeline, as we will describe in detail in Chapter 13. Some programming, at least in

6 'Author' isn't a verb, of course, but the usage is too well established to quibble over.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 65
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

a scripting language, is usually still required, if interactivity is to be provided.

The upshot of all this is that the making of a multimedia production can involve a host of software tools and skills, some of them special to multimedia, others belonging to existing disciplines. A full mastery of any one of the tools required may take years to acquire — Photoshop ping can be a career in itself, for example — and each demands a different kind of talent. It is a rare artist who can both write good programs and create high quality images, or direct videos, for example — so professional quality multimedia production will generally require a team. Which is not to say that one person working alone cannot produce multimedia, just that, in most cases, the result is unlikely to be of professional quality. This might be perfectly acceptable in some contexts: personal Web pages probably ought to look amateur, in a sense; scientific visualization is not judged by the same criteria as science fantasy creations; diagrams are judged differently from works of art, and corporate presentations have their own aesthetic. At the low end of the multimedia software market there are packages intended to make multimedia production, especially Web page design, accessible to non-specialists. Such packages hide technical details, such as HTML tags and JavaScript code, and usually provide 'wizards' or 'assistants' that guide the production process. Collections of clip art, basic animations and other prefabricated media elements are available, so simple multimedia productions can be produced quickly with the minimum of effort. They all come out looking much the same, of course.

An essential part of our definition of multimedia is the condition that the combined media be sufficiently well integrated to be presented via a unified interface, and manipulated by a single computer program. This is in complete contrast to the situation we have just described with respect to multimedia production, with its multitude of different software tools and team input. The key to integration is a framework that can accommodate a multiplicity of media and present them to the user. Three distinct approaches can be distinguished (although, as we will see, elements of these approaches can be, and are, combined).

The World Wide Web epitomizes one approach: define a format (in this case, a markup language, HTML or XML) that can accommodate different media, and view it using a dedicated browser. Here, all multimedia productions are collections of Web pages, which are

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 66
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

basically text, but usually contain embedded graphics and other media elements. Any Web page can be displayed by a sufficiently up-to-date Web browser.

The second approach is to define an architecture, comprising a format that can contain different media types, together with an API (Application Programming Interface) that provides a rich set of functions to manipulate data in that format. The prime example here is QuickTime. A QuickTime movie can contain video, sound, animation, images, virtual reality panoramas, and interactive elements, all synchronized and coordinated. The QuickTime API provides functions for playing and interacting with movies, as well as for editing them. Any program can use the API to provide multimedia functionality; QuickTime movies can be played by the QuickTime player that is distributed with QuickTime, in the same way that a Web page can be displayed in a browser, but they can also be emdedded in word processor documents, or within games, or any special-purpose program that needs to manipulate multimedia as well as performing some other tasks. One can imagine adding code which interprets HTML to any program, of course, but since QuickTime is implemented at the system level and is available to any program, the effort required to add QuickTime support is minimal in comparison.

The third approach is quite the opposite to the previous one: deliver the multimedia production in a 'stand alone' form, that needs no additional software to be used. Director provides a well-known example of this approach: movies may be saved in the form of a 'projector', which can be delivered on a CD-ROM, for example, and will play on any user's machine (of the right architecture) without their needing Director itself, or any browser or other movie-playing program.

Figure 2.11 summarizes these three approaches, and identifies some of the many applications that are available for each part of the multimedia production process. Although we have shown the process as one of production followed by immediate consumption, you should realize that in between these stages a multimedia production may be transmitted over a network, stored in a database, or written to an offline delivery medium. Conversely, although our examples show authoring systems that are used to combine media prior to delivery, the combination may occur 'on the fly' in response to user input. Finally, it should be borne in mind that these three approaches are not necessarily separate or incompatible.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 67
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

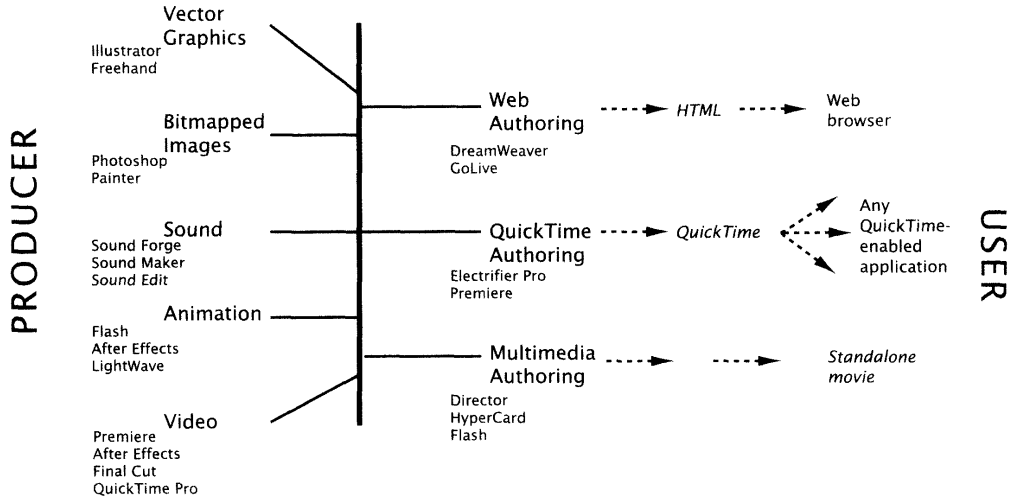


Figure 2.11
Multimedia production and delivery

QuickTime is frequently used purely as a video format with synchronized sound, for example, and so QuickTime movies can often be found embedded in Web pages or in Director movies. Similarly, Director movies can be saved in a form called Shockwave, and embedded in Web pages to be delivered over the Internet and played inside a browser. In order to accommodate these forms of embedded media, Web browsers must be extended with plug-ins, as we will describe in Chapter 13.

Networks

Modern computer systems rarely operate in isolation. Local area networks, usually employing some form of Ethernet technology, are used to connect computers within a small organization, or within a single site belonging to a large one. Local area networks are connected together by routers, bridges and switches to form internets. *The* Internet is a global network of networks, communicating via a standard set of protocols loosely referred to collectively as *TCP/IP*. Most of these networks are operated by commercial Internet Service Providers (ISPs) who lease access to other organizations via dedicated lines and routers, and to individuals via the telephone or cable TV networks. The growth of the Internet since the removal

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 68
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

of restrictions on its commercial use in 1991 has been more than adequately described and commented on elsewhere. Although, as we pointed out in Chapter 1, its ubiquity can be exaggerated, in developed countries the Internet is rapidly achieving the status of a public data network, analogous to the public communication network of the telephone system.

Networks, and the Internet in particular, offer valuable opportunities for distributing multimedia, but they also present formidable technical difficulties. In the case of the Internet, an additional factor to be considered is the presence of a largely technologically unsophisticated community of users, who have high expectations of technical quality based on their experience of television, videos, photography and print media. Where multimedia content delivered over the Internet attempts to compete with traditional media, as it is beginning to in news or sports coverage, it must provide some additional value (usually in the form of interactivity) to compensate for inferior picture and sound quality. This situation will, however, change as new, high-speed, access technologies come into widespread use.

For the present, most domestic Internet access is provided over a 'dial-up connection', with a modem and the telephone system providing a temporary connection between a user's computer and an ISP's network. The maximum bandwidth available through a modem is 56 kbits per second (kbps).[7] Many new computers are fitted with modems that conform to the V90 standard, which should provide this speed for data flowing 'downstream' from the telephone exchange to the modem, with a maximum of 33.6 kbps in the opposite direction, although factors such as noise and the distance between the modem and the telephone exchange make it rare for these figures to be achieved in practice. Speeds between 34 kbps and 48 kbps downstream are more realistic. Many older modems are still in use, providing speeds down to 14.4 kbps. For multimedia delivery it is normal to assume that it is unreasonable to expect to be able to use anything slower than 14.4 kbps, although it is considerate to provide some alternative to images, sound and video — usually a text substitute — for users with slow connections.

There is good reason to suppose that 56 kbps is the absolute maximum bandwidth achievable using analogue connections between the modem and the exchange.[8] Various new technologies avoid this analogue connection to make direct use of the digital telephone system's potentially higher bandwidth. *ISDN* (Integrated Services Digital Network) is based on the principle of combining multiple

7
Note that in this section, speeds are quoted, following normal usage in the networking literature, in *bits* per second, although elsewhere, following mainstream computing usage, we quote file sizes and so on in bytes.

8
But that's what was said about 33.6 kbps previously.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 69
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

	kbps	6KB text page	100KB image	4MB movie
Slow modem	14.4	3 seconds	56 seconds	37 minutes
Fast modem	56	1 second	14 seconds	9 minutes
Basic rate ISDN	128	< 1 second	6 seconds	4.3 minutes
T1 line	1544	< 1 second	1 second	21 seconds
Cable modem/ ADSL (typical)	6000	< 1 second	< 1 second	5 seconds
T3 line	44736	< 1 second	< 1 second	1 second

Table 2.1
Data transfer rates over the Internet

digital 'channels', each of which supports 64 kpbs. Its 'basic rate' can use two digital connections simultaneously to carry data at up to 128 kbps.[9] Although ISDN was a promising technology when it was introduced in the mid-1980s, non-technical factors have prevented its widespread deployment, and other technologies have since caught up with or overtaken it. Currently, *ADSL* (Asymmetric Digital Subscriber Line) appears to be the leading new method for access over existing copper telephone wires (albeit with different switching equipment). It provides speeds of up to 6.1 Mbps in the downstream direction, and up to 640 kbps upstream (hence 'assymmetric').[10] ADSL is being installed in homes in some areas of the United States in 1999, and European telecommunications carriers, including British Telecom, have announced their intention to make ADSL available in 2000. Another way of avoiding the restriction of the analogue connection between the home and the telephone exchange is to use the cable television network instead of the telephone network, in areas where cable TV is available. Cable modems can transfer data at rates from 500 kbps to 10 Mbps; devices capable of 30 Mbps will be available by 2000. Cable modems have an additional advantage of being permanently connected.

Commercial Internet users, especially those operating their own Web servers (see below), usually prefer to lease dedicated lines to provide a fixed connection to ISPs. *T1* and *T3* lines provide 1.544 Mbps and 44.736 Mbps, respectively. T1 and T3 lines are also used by small local ISPs to connect to the larger ISPs who are directly connected to the Internet backbone.

⇨ Although 'primary rate' ISDN, which utilizes up to 30 channels (in Europe) or 23 (elsewhere) is available, it is most often used for transferring files directly between, for example, a graphic design studio and a pre-press bureau, rather than for Internet access.

Table 2.1 shows the typical times taken to transfer different media elements over various types of connection, assuming the maximum speed is attained. Although the comparison is striking, it only tells

9
A third 16kpbs channel is used to carry control signals.

10
Other DSL variants offer lower but symmetrical rates (for example HDSL offers up to 1.5Mbps in both directions) or even higher assymetrical rates, up to 50 Mbps downstream).

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 70
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

part of the story. The connection between the user and their ISP is not always the factor that limits the speed of data transfer over the Internet. The capacity of the connections between ISPs' networks, and the computing power of the machines from which data is being downloaded may be just as important. It is quite common to find that a file download is not even using all the available speed of a V90 modem. However, anecdotal evidence suggests that both cable modems and ADSL can provide qualitative improvements in download speeds from powerful servers. For example, downloading the 25 Mbyte QuickTime movie of the trailer for *Star Wars: The Phantom Menace* from Apple's QuickTime site is typically reported to take less than three minutes using cable or ADSL — nowhere near the theoretical speed, but considerably better than any conventional modem. However, it is quite easy to see that the advantages of faster access between the user and the ISP can only be maintained if the connections between ISPs are upgraded proportionally as more users acquire faster access, and if data can be delivered by the machines at the far end of the connection at the same rate at which it can be transferred. Since data bus speeds are still considerably higher than network speeds, the latter condition is easily met for a single connection, but as the number of Internet users increases, popular sites will need to service many requests at once.

Local area networks provide much faster data transfer than the Internet. The most common technology for small organizations' LANs is 10 base T Ethernet, which provides 10 Mbps; increasingly, this is being superseded by 100 base T, at 100 Mbps, and, during 1999, the first Gigabit Ethernet equipment began to appear. This bandwidth must be shared between all the transfers taking place at any time, so the effective bandwidth between two computers on a LAN will be lower; it is still sufficiently high for multimedia applications that are presently infeasible over the Internet. In particular, video conferencing and other video applications, which can only be implemented on the Internet at very low quality, are possible using a high-speed LAN.

Clients and Servers

Online distribution of multimedia over LANs or the Internet is almost always based on the client/server model of distributed computation. In this model, programs called *servers* 'listen' on a communication channel for *requests* from other programs,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 71
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

called *clients*, which are generally running on a different machine elsewhere on the network. Whenever a server receives a request, it sends a *response*, which provides some service or data to the client. The requests and responses conform to a *protocol*, a set of rules governing their format and the actions to be taken by a server or client when it receives a request or response.

The most popular form of online multimedia delivery is the World Wide Web, whose implementation is an example of the client/server model. Web servers and clients communicate with each other using the *HyperText Transfer Protocol*, usually abbreviated to *HTTP*. HTTP is a very simple protocol, designed for the fast transmission of hypertext information, which is usually in the form of documents marked up using the HyperText Markup Language, HTML, which will be described in Chapter 8. However, the information in the World Wide Web is more properly described as hypermedia, and may include graphics, sound, video, MIDI and other sorts of data, and even programs.

HTTP provides communication between Web servers and their clients. A client first contacts a server, and then (usually) sends a request for a Web page. The identity of the server and the location of the file containing the Web page's data are normally extracted from a URL, the familiar 'Web address', such as `http://www.wiley.com/compbooks/chapman/index.html`, where `www.wiley.com` is the name of a machine running a Web server, and `compbooks/chapman/index.html` identifies a file in that machine's file system. The server responds by sending the contents of the designated file, if it exists, wrapped up in the form of an HTTP response with some extra information, such as the type of the data (HTML text, GIF graphics, sound, etc.) it is sending. This type information is specified in a format which will be described in the next section.

World Wide Web clients are usually browsers, such as Netscape Navigator or Internet Explorer, which allow us to access Web pages interactively. Despite the foregoing description, Web browsers are usually multi-protocol clients, which can communicate using other protocols, too. Nearly all browsers allow you to download files using the File Transfer Protocol (FTP), for example, although the interface to that protocol is integrated transparently into the Web browsing interface used for HTTP. Modern browsers also support real-time data streaming for audio and video using several special protocols for this purpose.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 72
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Web servers often run on dedicated powerful machines, usually under Unix, to enable them to cope with heavy traffic. A common arrangement is for ISPs to provide Web space on one or more of their machines running a server for the benefit of their customers; small companies and individuals wishing to put up personal Web pages can use the ISP's facilities. Larger companies maintain their own sites and servers. It is perfectly possible to run a Web server on a desktop machine, provided only a reasonable number of hits are expected — Watford football club's site uses a server running on an iMac, for example. The machine running the server must be permanently connected to the Internet, though, which will usually entail leasing a line, so it is not a viable option for most people.

The more powerful servers often augment their basic Web page serving function with interfaces to other programs running on the same machine. This allows them to generate Web pages dynamically, incorporating, for example, information retrieved from a database, and to perform computations based on information supplied through a form on a Web page. The Common Gateway Interface (CGI) is a *de facto* standard for such interfaces, but other proprietary mechanisms, such as Microsoft's Active Server Pages, and Apple's WebObjects, are increasingly preferred, largely because of efficiency and their tighter integration with database management systems.

Many local area networks are based on the same TCP/IP protocols that are used on the Internet, and often they use the same high level protocols, so that, for example, multimedia data can be served over a LAN using HTTP. A LAN organized in this way is often called an *intranet*. It allows familiar Internet clients, in particular, Web browsers, to be used with the private network to access information within the organization, but at higher speeds than those that can be attained over the public Internet. Alternatively, proprietary protocols may be used for multimedia applications operating over a LAN. These may provide services that are not available via the Internet, and may be more suitable for private use by a single organization.

MIME Types

The transmission of disparate types of media data across networks connecting heterogeneous computer systems calls for some way

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 73
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

of identifying the sort of data contained in a file or data stream. Individual operating systems have their own methods of identifying the type of a file. Most commonly, the extension of a file's name distinguishes the type: a file whose name ends `.JPG` on a Windows system is assumed to contain a JPEG image, one whose name ends in `.MOV`, a QuickTime movie, and so on. There is nothing standard about the mapping from content type to extension, though — Unix systems may use different extensions from Windows — and not all systems use extensions for this purpose. The MacOS has its own way of storing file type and creator codes with files, and not all media data is stored in a file. Some other means of identifying the content types is required in a networked environment.

MIME (Multipurpose Internet Mail Extension) is an extension to the Internet mail protocols that supports the inclusion of data other than plain ASCII text in mail messages. Some of its features have been adopted by HTTP, where they provide a simple and convenient way of specifying the type of data included in a server's response to a client's request for a resource. In particular, an HTTP response includes a MIME content type header, which takes the form:

`Content-type:` *type/subtype*

where *type* provides a broad indication of the sort of data, such as text, image, or sound, and *subtype* specifies the precise format, such as HTML, GIF, or AIFF. For example, HTML pages have the MIME content type `text/html`, while GIF images have type `image/gif`.

The available types are `text`, `image`, `audio`, and `video`, which have the obvious meanings; `model` for 3D model data, such as VRML; `message`, which indicates an email message, and `application`, which means binary data, including executable programs, that must be processed in some way: for example, PostScript would usually have the MIME type `application/postscript`, since it must be passed to a PostScript interpreter. (Conceivably, you might send PostScript with type `text/plain`, if you wanted the client to view the PostScript code itself instead of rendering it).[11] The range of subtypes is extensive (none more so than the subtypes of `application`) and supports most multimedia file formats, although some use 'experimental' subtypes, identified by the prefix `x-`, which, while not included in the list of supported MIME types maintained by the Internet Assigned Numbers Authority (IANA), are widely supported by Web browsers. For example, the MIME content type `video/x-msvideo` is recognized by Internet Explorer and Netscape

11 MIME also allows for arbitrary experimental types, which must begin with the characters `x-`, and for a `multipart` type, which has been used experimentally with HTTP for form data, but is more often used for mail messages with attachments.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 74
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Navigator as identifying AVI movies, although this format is not among the video subtypes registered with IANA. QuickTime movies, on the other hand, have MIME content type `video/quicktime`: since QuickTime is registered with IANA, the subtype does not have to begin with `x-`.

We will decribe how MIME types are used by Web servers and browsers in more detail in Chapter 13.

Standards

The International Organization for Standardization (ISO)[12] offers this description of standards:

> "Standards are documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics, to ensure that materials, products, processes and services are fit for their purpose."

12 ISO is not a mistyping of IOS, and it does not stand for International Standards Organization. ISO is not an acronym, but a play on words on the prefix iso- meaning 'the same'.

Since standards are agreements, we can assume that things that conform to the same standards are interchangeable. For example, an important set of ISO standards is concerned with screw threads. Any manufacturer's standard screws can be fitted in place of any other manufacturer's standard screws, which means that equipment that is assembled using standard screws is not tied to one screw manufacturer. In a market economy, this is supposed to lead to healthy competition. In practical terms, it means that the bankruptcy of one screw manufacturer does not imply that everything built with their screws becomes irreparable. However, only some standards have any legal status, so conforming to a standard is not something that anybody can usually be compelled to do. Nor does conforming to a standard necessarily mean doing something the best way; it just means doing something the standard way.

In multimedia, standards define interfaces, file formats, markup languages, network protocols, and so on, in precise, and usually formal, terms. The rôle of these standards is broadly analogous to that of the standards for screw threads. If we have a standard file format for image data, for example, then we can incorporate images into a multimedia production without having to worry about which

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 75
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

program was used to prepare them originally. Similarly, standards for all the other media types enable multimedia authoring systems to be constructed independently of the applications used to prepare the individual media elements. The alternative would be for each manufacturer of multimedia software to produce a system that could only use its own formats. Whereas this may be attractive to some manufacturers, such closed systems are unacceptable in a world where several different hardware and software platforms are in use, communicating with each other over networks; open systems that can accommodate data from any platform or source are needed.[13]

13
We use the word 'open' in a general sense, without implying the free availability of source code required by adherents to the Open Software credo.

Standards are of particular importance in networking, precisely because a modern network should be capable of connecting different makes of computer running different operating systems. It is only by conforming to standards for protocols and for electrical connections agreed to by all the manufacturers that this can be possible. Because modern networks cross national boundaries, and carry data all over the world, international standards are essential. In general, international standards are increasingly seen as being important because they facilitate global trade.

Three organizations are involved in making international standards that are relevant to multimedia: ISO, the International Electrotechnical Commission (IEC), and the ITU (International Telecommunication Union). ISO takes responsibility for formulating standards in all technical fields except electrical and electronic engineering, which are the responsibility of the IEC. Information technology defies categorization, and is dealt with by a joint ISO/IEC technical committee. ISO works through the national standards bodies of its member countries — ANSI (the American National Standards Institute) in the United States, BSI (the British Standards Institution) in the United Kingdom, DIN (Deutsches Institut für Normung) in Germany, and so on — who administer the committees that formulate standards, with ISO itself largely operating as a coordinating agency. The IEC operates in a similar way with the relevant national bodies responsible for electrotechnical standards.

Whereas ISO and the IEC are non-governmental agencies — commercial companies, in fact, albeit of a rather special nature — the ITU is an agency of the United Nations.[14] It has a more regulatory function than the other two international standards bodies. For example, it is the ITU that allocates frequencies to radio services; these allocations must be followed to prevent interference, so they have a mandatory

14
Although its history goes back much further than the UN's, to 1865.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 76
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

status according to international treaty. ITU standards covering video formats and telecommunication are those most relevant to multimedia.

Although there is broad agreement on the desirability of standards, the process of agreeing standards through these official bodies is a fraught and often a long drawn out one. Since standards will only be observed if there is a consensus among the concerned parties, a great deal of politics and compromise may be involved in the production of a standard. The major standards bodies require extensive consultative procedures to be followed, designed to ensure that a standard has the broadest possible support, before a draft document can be endorsed as a full standard. These procedures work well for such things as screw threads, where a standard may be expected to remain relevant for very many years after it has been created, but in the rapidly changing environment of computers, networks, and multimedia, standards are often obsolete before they have passed through all the necessary stages and national and international committees. Furthermore, international standards bodies derive some of their income from the sale of standards, and are therefore reluctant to make their documents freely available. This causes some resentment in the computing community, which has become used to free documentation on the World Wide Web — and the high prices charged for some standards put them out of the reach of small software companies. As a result, semi-formal standards and *ad hoc* arrangements play a greater rôle in these areas than they do in more traditional fields of engineering.

Internet standards are a paradigm of this semi-formal standardization. Since the Internet is, by definition, an open network architecture, it relies on standard protocols to enable different networks to be interconnected. The Internet grew out of Arpanet and NSFNET, which had some degree of central administration; the TCP/IP protocols could be imposed in the early days of internetworking and were inherited as the basis of the Internet. Responsibility for the further development of protocols and for administering information required for the protocols to operate now resides with the Internet Architecture Board (IAB) and its subsidiaries, including the Internet Engineering Task Force (IETF), which deals with technical development, and the Internet Assigned Numbers Authority (IANA), which registers MIME types, language codes, and so on. These bodies have no formal standing outside the Internet community, and no statutory powers.[15] Similarly, the

15
IETF documents, including Internet standards, are almost diffidently called 'Requests for Comments'.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 77
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

organization responsible for defining World Wide Web standards, the World Wide Web Consortium (W^3C), has no official status but its Recommendations are treated as standards. As you would expect, these bodies make use of the Internet as a means of disseminating standards, both after they have been adopted, and during the standardization process, thereby providing an opportunity for a wide audience to comment on drafts and proposals.

The advantage of such an *ad hoc* approach to standards is that it accommodates rapid change. The disadvantage is that manufacturers feel less compunction in ignoring, adapting, or extending standards. There is a fine line between such behaviour and legitimate experimentation aimed at advancing a standard. This is illustrated by the history of HTML, where the two main competing browser companies, Netscape and Microsoft, each implemented their own extensions to HTML 2.0, leading to incompatibilities, but ultimately most of the extensions were incorporated into later versions of the HTML standard.

Sometimes, standards are established without the intervention of standards bodies of any sort. One company's product may come to dominate the market to such an extent that it becomes a standard in all but name. The PostScript page description language and the QuickTime multimedia architecture are examples. In both cases, their perceived technical superiority, possibly reinforced by the marketing muscle of Adobe and Apple, respectively, has caused most rivals to be abandoned, leaving them as standards by default. In some cases, a *de facto* standard of this sort may be more widely used than a competing official standard. For example, at the time of writing, the W^3C is developing a standard for vector graphics on the Web, SVG, but, while this has not been implemented, Macromedia's Flash format is supported by the major browsers, and is in wide use as the 'standard' Web vector format.

Further Information

Sampling and quantization are described in most books on image processing or sound, e.g. [Bax94] and [Poh95]. Hardware and software develop at such a rate that it is usually best to consult trade journals and magazines for the latest developments. We return to the topic of multimedia and networks in Chapter 15. [ISO]

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 78
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

gives ISO's perspective on standards, while [Bra96] describes the IETF's standardization procedure for the Internet.

Exercises

1. Identify three phenomena in the natural world that exhibit continuous change.

2. The second hand on the watch in Figure 2.1 appears to be broken in one place. Explain why it looks like this. (Hint: The picture was made by scanning an actual wristwatch.) What can you deduce about the way the second hand moves?

3. Is the sampling rate of 44.1 kHz used for audio CDs adequate to reproduce musical sounds accurately? Justify your answer.

4. Suppose a piece of film depicting a moving stagecoach is shot at 24 frames per second, and that the wheels are rotating at such a speed that, when the film is projected at 24 frames per second the wheels appear to move backwards. What would you expect to see if the same film is projected at (a) 12 frames per second; (b) 30 frames per second; (c) 60 frames per second?

5. Digital representations of sounds and images are sometimes accused of being 'limited' (e.g. by the number of quantization levels or the sampling rate), and consequently not to be 'true' representations. Are analogue representations also limited? Explain your answer and give examples.

6. Prove (at least to your own satisfaction) that, if we double the number of bits used to hold a quantized value, then we square the number of quantization levels.

7. Draw up a list of the hardware and software you would recommend for equipping:

   (a) A small company specializing in Web page design.

   (b) A community college or Further Education college laboratory for introductory courses on multimedia.

   (c) A public library's IT facility for community multimedia access.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 79
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

If possible, talk to people working in the relevant sectors to get an idea of realistic budgets, and try to fit your recommendations within them.

8. Which of the following types of media production are likely to be succesful on a Web site, given current technology, and which may cause problems?

   (a) A live sports broadcast, (i) sound only, and (ii) sound and video.

   (b) The personal home page of a stamp collecting enthusiast.

   (c) A video clip from a pop promo.

   (d) An art gallery's catalogue of its current exhibition, with text and still images.

   (e) An interactive computer game with high quality graphics.

   (f) A display of up-to-the-minute share prices for distribution to market traders.

   (g) A hi-fi live broadcast of a classical music concert.

   Explain each of your answers. For those productions that you do not consider likely to be succesful, what high-end equipment or presently envisaged technological advances would be needed to make them feasible?

9. The ISO standard ISO 216 defines a collection of standard paper sizes, including the A sizes, A0, A1, A2, and so on, of which A4 is the most popular for office use. Paper made to this standard is used almost everywhere in the world except in the United States. What problems does this exception cause for hardware and software manufacturers and users? What factors prevent the adoption of ISO standard paper sizes in the United States?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 80
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

3 Introduction to Computer Graphics

We use the term *graphics* in a broad sense to refer to the software and hardware technologies used in a computer system to create, modify and display still images stored in a digital form.

Graphics in this sense is of fundamental importance in multimedia, not only because it allows us to generate and display still pictures, but also because it underlies the display of moving pictures and text. Graphics should not, therefore, be considered as just one of the media that make up what we call multimedia, in the sense that sound is, for example. Rather, it is the enabling technology for all the visual elements of multimedia. To understand how it fulfils that rôle, we need to examine how it works in isolation first. In other words, we need to consider the production and display of still images.

Digital images may originate in a number of different ways. They might already exist in some non-digital medium, and be digitized by a scanner, or they might be captured from the external world

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 81
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

in digital form by a digital camera or video frame grabber. Other images might be created on a computer system, by an artist, designer or illustrator using a graphics package, or they might be built up by a programmer using a graphics language. Extensive collections of digital images are available on CD-ROM and on commercial and amateur Web sites. Finally, images might be generated by a computer program operating on some data, mapping it to a simple visual representation such as a pie-chart or a more elaborate visualization, such as a simulated picture of the wave structure of electrons in a solid.

There is a long history of images being made and used as art, entertainment, information, inspiration, devotion, titillation, education, amusement, decoration and communication. In a way, this makes still images the easiest of media: we can draw on centuries of experience when creating images, and, at least within a specific cultural context, we can reasonably expect our audience to understand an extensive range of cultural and visual conventions, such as perspective or caricature. This very familiarity can also make images difficult to deal with in a digital system, because it raises expectations of what can be achieved, which the limitations of the systems available to us often frustrate.

Although some painting programs can simulate the effects of real art materials to a remarkable extent, and high quality scanners can capture much of the detail and subtlety of an original, in the final multimedia production your images will almost certainly be displayed on a low resolution monitor which cannot match even the quality of reproduction provided by photographic methods in glossy magazines. Worse, the available range of colours may be limited, and will be reproduced differently on different systems. It is necessary to understand these limitations, so that you can take appropriate steps to ensure that your artwork looks as good as it can do within them.

Graphic elements of multimedia productions will usually be delivered on CD-ROM, DVD or over a network, often the Internet. However, digital images are also widely used for print media. It may be necessary to adapt such images for use in multimedia or, contrariwise, to allow for the printing on paper of images originally made for display on a monitor. To do this effectively, it is necessary to take account of the different characteristics of displays and printers, particularly in respect of colour, a topic we will examine in detail in Chapter 6.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 82
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

There is no doubt that, ultimately, display technology will improve to a point where much higher image quality is achievable. It will always remain the case, though, that an image works — conveys its meaning, evokes its response — differently when it is in a digital form than it does when it is reproduced in print, or framed and hung on a wall. A notable feature of the World Wide Web's shift from a text-based medium to a heavily graphical medium has been the way traditional graphic designers failed to appreciate this difference, and tried to transplant established idioms from print-based design on to the Web. Often the results were cumbersome and unreadable pages that took many minutes to download, looked terrible on most ordinary monitors, and served only to obstruct users trying to find information. Latterly, the Web has begun to develop its own visual vocabulary, which takes account of its limitations and takes advantage of its particular capabilities.

The use of icons as components of graphical user interfaces provides one example of the way in which images can be put to new use in a computer system. Another example is the way in which visual representations of data can be generated and displayed to help make large quantities of information more readily comprehensible, as we mentioned in Chapter 1. Such visualizations depend on computers to perform the necessary calculations to generate the image from the data; they are therefore designed with the limitations of display technology in mind.

Vector Graphics and Bitmapped Graphics

The display of images is ultimately controlled by some program: a dedicated picture-displaying application, an image editor, or a Web browser, for example. Monitors display pictures as a rectangular array of *pixels* — small, usually square, dots of colour, which merge optically when viewed at a suitable distance to produce the impression of continuous tones. To display an image on the monitor, the program must set each pixel to an appropriate colour or shade of grey, in order that the pattern of pixels on the screen produces the desired image. The low level operations required to set pixel values are usually performed by a graphics library, which communicates with the display hardware, and provides a higher level interface to the application program.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 83
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

A graphics application program must somehow keep an internal model of the image to be displayed. The process of generating a pattern of pixels from a model is called *rendering*. The graphic model will usually take the form of an explicit data structure that holds a description of the image, but it may be implicit in the sequence of calls to the graphics library which are made as the program executes. Where picture data must be persistent (i.e. must live on after the execution of the program, so that the image can be displayed again at a later time, possibly by a different program), a similar model must be kept in a file. The sequence of events for displaying an image then begins with a program reading an image file, from which it constructs an internal data structure corresponding to the image description in the file; the program then renders the image for display by calling functions from a graphics library, supplying arguments derived from its image model.

⇨ Where images are being generated as visualizations of data, another level of modelling may be present. For example, if a program is generating weather maps from satellite data, then it must have a model of the visual elements of the map — isobars, weather fronts, the underlying geographical features — but it must also maintain a mathematical model of the state of the atmosphere, based on the data, from which the map model is generated.

It is usual to distinguish between two different approaches to graphical modelling: *bitmapped graphics* and *vector graphics*.

In bitmapped graphics, the image is modelled by an array of pixel values. Where it is necessary to emphasize the distinction between these stored values and the physical dots on a display screen we will call them *logical pixels*, and the latter *physical pixels*. In the simplest case, the logical pixels correspond one-to-one to the physical pixels: the model is a map of the displayed image. More generally, the model may be stored at a different resolution from the displayed image, so that some scaling has to be applied to the logical pixels before the image is displayed; the model may also describe a larger image than that to be displayed, so some clipping will have to be applied to extract the desired part for display. Scaling and clipping are the only computations that need to be performed to display a bitmapped image.

In vector graphics, the image is stored as a mathematical description of a collection of individual lines, curves and shapes making up the image. Displaying a vector image requires some computation to be performed in order to interpret the model and generate an array

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 84
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

of pixels to be displayed. For example, the model will represent a line by storing its end-points. When the model is rendered for display, the co-ordinates of all the pixels lying on the straight line between those end-points must be computed so the pixels can be set to the appropriate colour. For persistent storage in a disk file, such a model is often implemented as a program in a graphics language, for example, PostScript or PDF.

⇨ Neither of the terms 'vector graphics' or 'bitmapped graphics' is entirely accurate. A more accurate term for what we are calling vector graphics would be 'object-oriented graphics', were it not for the potential confusion caused by the meaning of the term 'object-oriented' in the field of programming languages. Bitmapped graphics does not use *bit*maps (except for purely monochrome images) it uses pixel maps, or *pixmaps* for short, but the term 'pixmapped graphics' has never acquired any widespread currency. You will sometimes see the name 'graphics' reserved for vector graphics, and 'images' used to mean bitmapped images, but this usage can cause confusion with the colloquial meaning of these words. Throughout this book, we will stick to the terms introduced in the preceding paragraphs, and hope you will not be misled.

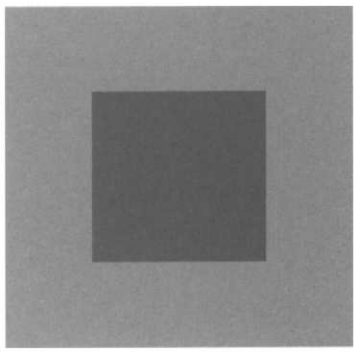


Figure 3.1
A simple picture

There are profound differences between vector and bitmapped graphics. It should be evident that they will make different demands on your computer system: a bitmapped image must record the value of every pixel, but a vector description can be much more compact. For example, consider an extremely simple picture consisting of a magenta square inside a green one. (See Figure 3.1, although it lacks the colour.) The complete picture is 45mm square; if it was stored with 72 logical pixels per inch (meaning that it could be displayed on a 72dpi monitor at its natural size without scaling), then it would be 128 pixels square, so its bitmap would consist of $128^2 = 16384$ pixels. If we assume that the intended destination is a consumer-level monitor capable of displaying 256 colours at a time, then we need 8 bits to distinguish the possible colours (see Chapter 6), which means that each pixel occupies one byte, and the entire image occupies 16 kilobytes of memory. The same picture, on the other hand, could be stored in the form of a description of its component rectangles and their colours. One possible format would be a short program in the PostScript page description language, such as the following, which occupies a total of just 78 bytes:

```
0 1 0 setrgbcolor
0 0 128 128 rectfill
```

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 85
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

```
1 0 1 setrgbcolor
32 32 64 64 rectfill
```

The first line sets the 'paint' colour to green, the second draws the outer square and fills it with that paint; the second pair of lines similarly construct the inner, magenta, square. Whereas displaying the bitmapped image only requires the pixels to be copied onto the monitor, displaying the PostScript version requires some software, in this case a PostScript interpreter, to translate it into a displayable form. This not only slows down the display of the picture, it also relies on a PostScript interpreter being available on the computer where the image is to be displayed.

The sizes of bitmaps and vectors are affected by the content of images in different ways. The memory requirement for any 45mm square, 256 colour bitmapped image at a resolution of 72 pixels per inch is 16 kbytes, no matter how complicated the image may be. In a bitmapped image we always store the value of every logical pixel, so the size of the image and the resolution at which it is stored are the only factors determining the amount of memory occupied.[1] In a vector representation, we store a description of all the objects making up the image; the more complex the picture, the more objects there will be, and the larger the description that will be required, but since we do not actually store the pixels, the size is independent of any resolution.

At least as important as the technical differences between the two different types of graphics are the differences in what you can easily do to your images using each approach. Although many image editing programs now allow you to combine vector and bitmapped graphics, traditionally a distinction has been made between *painting* programs, which operate on bitmaps, and *drawing* programs, which work with vector representations. Even though the distinction is less clear-cut than it used to be, it is still the case that packages such as Illustrator or Freehand are primarily intended for creating and editing vector graphics, and only provide limited facilities for dealing with bitmaps, while packages such as Photoshop and Painter, while providing extensive support for bitmap manipulation, either offer little support for vector graphics, or provide a poorly integrated drawing sub-system.

Figures 3.2 and 3.3 show two flowers; the first is a vector drawing of a poppy, made in Illustrator, the other is a bitmapped image, scanned from a painting of an iris, executed in gouache. (Plates 3 and 4 show the same two flowers in colour.) The bitmap

1 Unless we apply compression to it — see Chapter 5.



Figure 3.2
A vector poppy

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 86
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046



Figure 3.3
A bitmapped iris



Figure 3.4
Transforming a vector image



Figure 3.5
Applying effects to a bitmap

captures the textures of the original picture and provides a good reproduction of the continuous tones and translucent quality of the paint. The vector drawing has a quite different character, with clearly delineated shapes, made out of a collection of smooth curves, filled in with flat colour.

⇨ These examples have been deliberately chosen to emphasize the differences between the two types of image. A good illustrator working with a professional drawing package can produce much more subtle effects, but vector images are always based on the same elements of filled shapes.

Figures 3.4 and 3.5 demonstrate the difference between the two formats in another way. With the vector representation, it is easy to select the individual shapes — petals, stem, stamens — and move, rotate or otherwise transform them independently. Each element of the picture retains its identity and can be edited as an object because the position and attributes of each object are stored in the image model. To achieve a similar effect with the bitmapped image would require a painstaking process of masking out the required pixels and then re-touching the gap where the petal had been removed, since this image is just an array of pixels, with no underlying model to indicate which belong to the stem, and which to the petals. As a result, editing of parts of the image must be performed by selecting areas, either by painstakingly drawing round them or semi-automatically (and somewhat unreliably) by searching for tonal discontinuities or areas of similar colour. On the other hand, applying a special effect, such as distortion or blurring, to the bitmapped image is simple, whereas producing the same distortion on the vector image could only be done by first transforming it to a bitmapped format, since such effects transform each pixel, possibly using the value of its neighbours, but independently of whether they belong to the same object.

Another major difference between vector and bitmapped graphics is the way they behave when scaled or re-sized. If a bitmapped image is to be displayed at greater than its natural size, each logical pixel must be mapped to more than one physical pixel on the final output device. This can be achieved either by multiplying up the logical pixels, effectively increasing their size (for example, to double the linear dimensions, each pixel value in the model should be used to set a block of four pixels on the display), or by interpolating new pixels in between the stored ones. In either case, the effect will usually be a readily perceptible loss of quality. Since a vector image

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 87
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

consists of a description of the component shapes of the picture, not the values of pixels, scaling can be performed easily as a simple mathematical operation, before the pixel values are calculated. As a result, curves, for example, will remain smooth, no matter how much a vector image is blown up, whereas they will become jagged or blurred if a bitmapped image is scaled up. Figures 3.6 and 3.7 illustrate this effect, being details of the two flower images scaled up by a factor of 8. The outlines of the poppy remain smooth, whereas the iris has become coarse, and the blocks of the original pixels are clearly visible, especially along the edge of the petal.

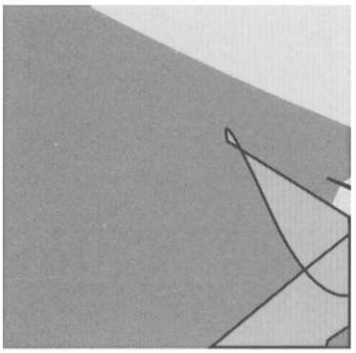


Figure 3.6
Scaling a vector image

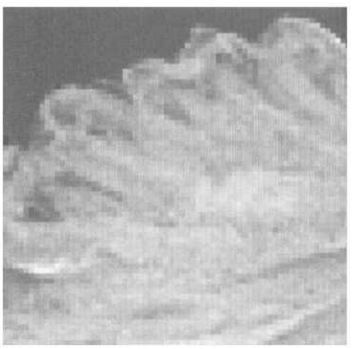


Figure 3.7
Scaling a bitmap

⇨ Related problems arise when bitmapped images are displayed on devices with different resolutions, since either they will be the wrong size, or they will exhibit the same loss of quality as occurs when they are scaled. This is more of a problem when images are being prepared for printing than it is with multimedia, since in the latter case we know that our images are going to be displayed on a monitor. Monitor resolutions do vary: whereas standard Macintosh displays have a resolution of 72 dpi, PC monitors usually use a resolution of 96 dpi, while multiple scan displays may provide a resolution of as much as 115 dpi at their higher settings. As a result, images prepared using such a display at a high resolution will appear larger on an ordinary PC or Macintosh. (Users of multiple scan displays will be familiar with the way in which all their desktop icons change size when they change the resolution.) This is usually considered acceptable, but should be borne in mind by designers.

The terms 'drawing program' and 'painting program' introduced earlier express the difference between the visual characteristics of vector and bitmapped graphics. A drawing program lets you build up a vector image out of individual curves, lines and shapes, so the result usually has much of the character of a pen and ink illustration, an air-brushed painting, or a technical diagram: shapes are defined by outlines, and colour is applied to the regions defined by those shapes. A painting program allows you to make a wide range of different marks, and apply colour to arbitrary areas of the image; modern painting programs do a pretty good job at simulating the appearance of natural media such as charcoal, pastel, watercolour or oil paint and the interaction of those media with textured and absorbent supports, such as coarse watercolour paper or canvas.

Drawing and painting programs use different sets of tools to perform the operations that are most appropriate to vector graphics and bitmapped images, respectively. Drawing programs have

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 88
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

tools for constructing shapes such as rectangles and ellipses, and for drawing lines and curves, selecting objects, and moving and transforming them by scaling, rotation, reflection and skewing. Painting programs have a quite different set of tools, including brushes for making marks in a variety of styles, colour correction and retouching tools and filters for altering images, and selection tools for picking out areas and groups of pixels.

Painting programs generally offer more expressive possibilities, but at the expense of high memory requirements and scalability problems. For many applications, such as the display of data for scientific or business purposes, technical illustration, and some sorts of graphic design, the expressiveness of painting programs and bitmapped graphics is unnecessary, or even positively undesirable, and vector graphics produced by drawing programs will be preferred.

Memory requirements, the visual characteristics of the image produced, and the possibilities for transformations and effects might influence your decision as to which format to work with. Another important factor is the source of your image. Scanned images, screen shots, photographs from a digital camera, and captured video frames are all inherently bitmaps, because of the way the hardware from which they originate works. Accordingly, for the manipulation and re-touching of photographs and for most pre-press work, painting programs are the only possible choice. Charts, diagrams, and other data visualizations generated by a program from data usually, but not invariably, use vector graphics. Artwork made on a computer can be in either form, with the artist's personal preferences, together with the factors mentioned above, and the availability of suitable software, determining the choice.

Combining Vectors and Bitmaps

The different qualities of vector and bitmapped graphics, deriving from the fundamentally different image representations each employs, make them suitable for different tasks. It is not uncommon to find tasks requiring a combination of these qualities that call for images containing elements of both types. For example, a graphic designer might wish to use a scanned image as a background over which drawn (vector) shapes are arranged. There are several ways in which this can be achieved.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 89
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

The first is to transform vectors into bitmaps, or *vice versa*. Hence, our designer might prepare their drawn elements in a vector graphics progam, turn the drawing into a bitmap, and composite that with the scanned background in a bitmapped image manipulation program. It is relatively easy to turn a vector graphic into a bitmapped image. The process of interpreting the vector description, known as *rasterizing*, can be accomplished using the same algorithms that are used to display the image on a monitor. The rasterized image loses all its vector properties, though — it ceases to be resolution-independent, so a resolution must be chosen when the rasterization takes place, and the individual shapes can no longer be selected and transformed, they just become areas of pixels. However, as a result they can be treated to the full range of bitmapped effects.



Figure 3.8
A vectorized bitmap

Going in the opposite direction, from pixels to vectors, is more problematical. It requires software to identify the boundaries of shapes within the image, then to approximate that boundary using the available sorts of curves and lines, and to colour them in appropriately. When the original image features subtle tonal graduations, and soft outlines, the process is not likely to be very successful. Figure 3.8 shows the result of an attempt to vectorize the bitmapped flower image from Figure 3.3. (It was done with Flash.[2]) The resulting vector image is made out of many small curves and shapes, which means that it is neither as compact nor as easy to edit as an image made using vector graphics from the start. (In fact, the file containing this particular vector image is bigger than the bitmapped version at screen resolution it was traced from.) Resizing, rotation, and other transformations that can be performed naturally and simply on vectors can be applied to the vectorized image.

2
And to be fair to Flash, we could have adjusted some parameters to make a better job of it, at the expense of creating a picture made out of an enormous number of tiny vectors.

⇨ Vectorization can be used in other ways than simply as a means of changing the representation of an image. It can be used in a controlled fashion to generate a starting point for a new vector image. Figure 3.9 shows the result of applying Illustrator's autotrace tool to the iris painting. This tool merely traces the outline of a shape, producing the path shown here. While this is barely recognizable as a version of the original, it could now be worked on inside Illustrator as part of a new composition based on the organic shape of the flower.

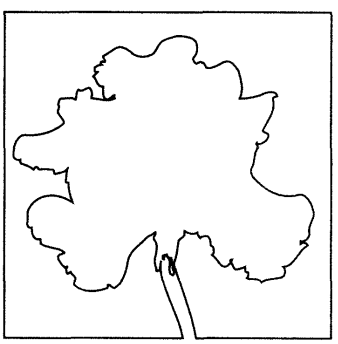


Figure 3.9
An autotraced bitmap

The autotrace tool will find different shapes depending on where it is applied. Figure 3.10 shows the paths produced by deliberately choosing parts of the image to produce a collection of curves that

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 90
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046



Figure 3.10
A more finely autotraced bitmap

make a better approximation to the iris. Again, these could be used as the basis of a new drawing, retaining the shapes but exploiting the capabilities of the vector tools.

Most drawing programs allow you to import bitmaps without vectorizing them. A bitmap imported in this way is treated as an indivisible object; it can be moved, and some transformations may be applied to it (although they might affect the quality of the image). However, it cannot be broken into its component shapes, the way a vectorized image or any shape drawn in vector form can be, and certain filter effects that rely on being able to change individual strokes cannot be applied to it. Commonly, drawing programs allow pointers to bitmapped images to be imported instead of making a copy of the image itself. This means that the bitmap can be edited in a painting program, after it has been incorporated into a vector image, and the changes will be seen in the vector image.

Sometimes it is not actually necessary to incorporate bitmaps into vector graphics; all that is wanted is for the vectors to have the appearance of bitmapped images. Until quite recently this could only be achieved by rasterizing and retouching in an image editor. Nowadays, some drawing programs — MetaCreations' Expression and version 8.0 of Illustrator, for example — allow 'brush strokes' to be applied to the lines and curves making up vector shapes, so that they appear to have been made with natural media, such as watercolour or pencil; calligraphic effects, such as a variation in the thickness of a line depending on its direction, can also be achieved.[3] This application of a natural media appearance to vector strokes is quite distinct from using a painting tool to draw a smooth curve, for example. In the latter case, the pixels on the path following the brush's movement are coloured to simulate the effect of a painted or drawn line; in the former case, the path is stored in the usual vector form, as the defining parameters of a collection of curves, and the associated appearance is added to it when it is displayed. The path retains all the desirable qualities of vector graphics: it can be transformed, altered, and displayed at any resolution, but it does not have the uniform appearance normally associated with vector graphics. Perhaps most interestingly, the brush stroke applied to a path can be changed, so the appearance of the marks can be altered without re-drawing the path. As far as the appearance of the resulting artwork is concerned, these effects blur the distinction between drawing and painting programs, but internally the strokes are still being applied algorithmically to paths stored in vector form.

3
Similar technology is used in Painter to allow vector strokes to be used in a bitmapped image editor, with all the natural media from that program's bitmapped world available to them.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 91
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Layers

The arrangement of artwork on *layers* is an organizational device that is common to both vector and bitmapped images. Since the introduction of the concept of layers in Photoshop 3, it has become one of the most significant ways in which digital technology has affected how artists, designers and illustrators work. As we will see in Chapters 10, 11 and 14, the intuitive appeal of the layer metaphor has led to its adoption in other media, too.

A layer is often likened to a digital version of a sheet of clear acetate material, like an overhead projector transparency. You can draw or paint on parts of the layer, leaving some of it transparent. An image can be constructed by stacking layers on top of each other; where a layer is transparent, the layer below it shows through. This may not sound very exciting — you can always draw things on top of other things — but a layer allows you to collect together part of an image and treat it as a unit.

An immediate consequence of this ability is that it provides a way of distinguishing objects in a bitmapped image. Normally, as we have shown, if you make a picture of a flower as a bitmapped image, there is no discrete object corresponding to each petal, just areas of pixels. By placing each petal on a different layer, though, they can be moved or modified individually, much as the individual shapes making up a vector image can be. One specific way in which artists take advantage of the separation that layers allow is by using one layer as a background against which objects on other layers are superimposed. These objects can then be moved about over the background, until a satisfying arrangement is found. If they were not on separate layers, whenever an object was moved, it would be necessary to touch in the background where it had been. In a similar way, the use of layers makes it easy to apply effects to parts of an image, since effects can be applied to individual layers. Thus, for example, a background layer might be blurred so that elements on layers placed over it will stand out.

Layers facilitate a graphic style resembling collage. You will probably recognize the Layered Look in much graphic design that was produced in the 1990s.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 92
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

A different way of using layers is as digital tracing paper. For example, rather than try to vectorize an image like our iris, you might prefer to import it into a drawing program in its bitmapped form, and then create a new layer on top of it, on which you could draw with the vector tools, using the imported image as a guide. (Drawing programs usually allow you to dim layers so that you can more easily see what you are doing on another layer.) Once that task has been achieved, the guide layer can be deleted without affecting any other layer.

Yet another way of using layers is for experimentation. Layers can be reordered without affecting their contents, so that different stacking arrangements can be tried out. Layers can be duplicated, and the duplicates altered separately; any layer can be made invisible, so different versions of a layer can be displayed in turn, to see which is better. When a picture is finished, it is normal to delete any invisible layers, and merge all the remaining ones down to a single layer, since this uses less space.

We have likened layers to transparent sheets of acetate, but, as data structures inside a computer are not subject to the physical limitations of materials, they can behave in ways that acetate cannot. In particular, the degree of transparency can be varied.[4] By using layers that are only partially transparent, backgrounds can be dimmed. The precise way in which separate layers are combined may also be modified. The normal behaviour of layers is for the areas that are not transparent to cover any layers beneath. Sometimes, it may be preferable for these areas to be blended with lower layers instead, or dissolved into them. Transparency may be made conditional on the brightness of one or other of the layers, so that blending takes place below a threshold value, but above that value the superposed layer conceals what is below it.

Figure 3.11 is a simple illustration of combining layers. The starting point is the two photographs shown at the top, one an Alpine hillside in winter, the other a piece of woodland. These were scanned, corrected for tonal balance, and cropped to the same size. The light areas of the woodland scene were selected and filled with pure white. This modified image was then pasted as a new layer over the background of the hillside. The hillside is unaffected, being on a separate layer. At the bottom, we show these two layers combined, with different threshold values being used to control the layer blending. On the left, the lower level shows through, producing a slightly disturbing incoherent scene; this was achieved

4 In Photoshop you actually change the opacity, which is one minus the transparency.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 93
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

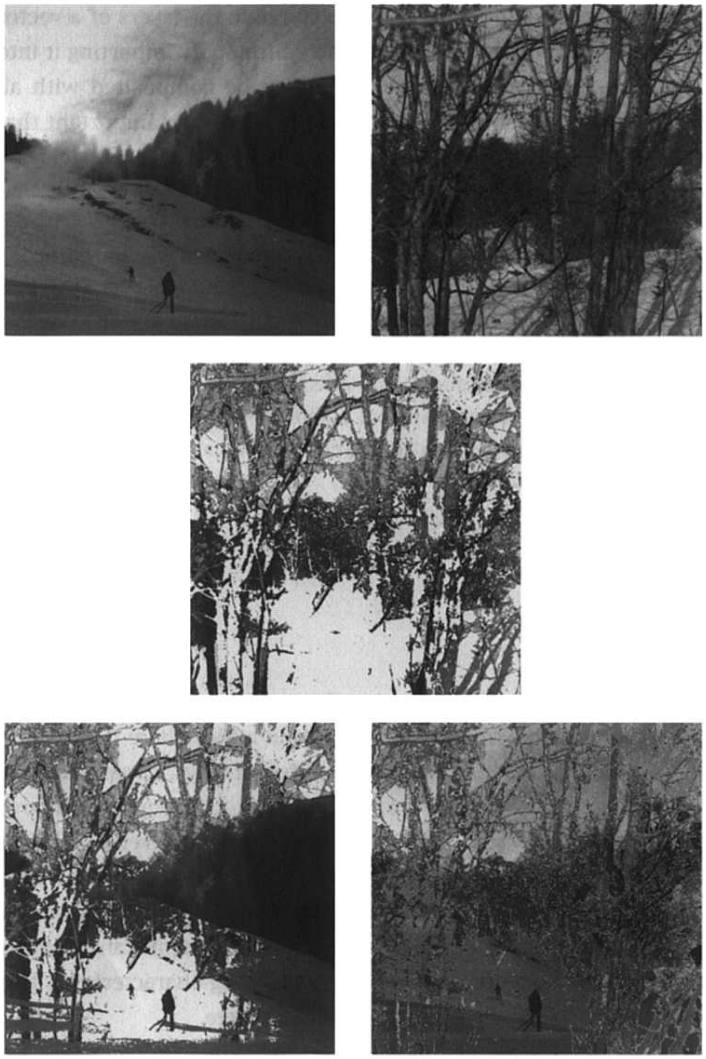


Figure 3.11
Producing artificial compositions with layers

by using a low white threshold and a high black one for the blending on the upper layer. On the right, by raising the black threshold for the background layer to quite a high value, the dark bulk of the hill on the right hand side of the picture has been effectively obscured, and the trees appear to be part of the same scene as the skiers — we have manufactured a new landscape.

Layer compositing of this sort can only be performed on bitmapped images, because it is done by computing the value of each pixel

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 94
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

individually. When it is necessary to combine the layers of a vector illustration, it must be converted into a bitmap by importing it into a painting program, where its layers can be composited with all the flexibility available. For this to be possible, it is important that the layers be preserved when the image is imported — only certain combinations of programs can do this correctly.

⇨ The layer metaphor has found such favour as a way of organizing images that it has been extended to incorporate effects as well as image elements. Photoshop's *adjustment layers* are described as being a layer through which you can look at the image through a medium that applies some effect, such as a tonal adjustment. In practice, this makes them a tool for applying effects without changing the pixels on image layers. They thus provide a safe means of experimentation.

File Formats

In order to preserve images and exchange them between programs, it must be possible to store image data in a file. There is considerable scope for encoding the data in different ways, compressing it, and adding supplementary information to it. Consequently, a large number of different graphics file formats has been developed.[5] As is the case with programming languages, most of these are only used by a limited circle of enthusiasts, for some specialized applications, or on particular platforms. There remains a significant number of different formats in wide use, with differing characteristics that make them suitable for different types of image.

5 The *Encyclopedia of Graphics File Formats* [Mv96] describes nearly 100, with more on its associated Web site.

For bitmapped images, one of the main differences among file formats is the way in which they compress image data. Bitmapped images can contain a large number of pixels, so they require large files, often occupying several megabytes per image at medium to high resolutions. In order to reduce the storage and bandwidth requirements of such images, data compression techniques are often applied to them. We will describe image compression in more detail in Chapter 5, but you need to be aware at this stage of the distinction between *lossless* and *lossy* compression. Lossless compression algorithms have the property that it is always possible to reconstruct the original data exactly from its compressed version; lossy algorithms discard some data — in the case of images, data

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 95
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

representing visually insignificant details — in order to achieve greater compression.

We also defer a full description of colour to a later chapter (Chapter 6) but note that one way of reducing the size of a bitmapped image is to restrict the number of different colours that it can contain. If an image uses at most 256 different colours, each pixel only requires a single byte, whereas if it is to be allowed the full range of millions of colours which most monitors are capable of displaying, three bytes per pixel are needed.

The advent of the World Wide Web has had something of a standardizing influence; although it does not specify any particular graphics file formats, the necessity for cross-platform compatibility has led to the adoption of certain formats as *ad hoc* standards.

The first of these is *GIF*, originally developed by CompuServe as a common format for exchanging bitmapped images between different platforms. GIF files use a lossless compression technique, and are restricted to 256 colours. One of this format's most useful features is that one colour can be designated as transparent, so that, if the GIF image is displayed against a coloured background or another image, the background will show through the transparent areas. This, in effect, allows you to produce images that are not rectangular. GIFs are most suitable for simple images, such as cartoon-style drawings and synthetic images produced on computers. They are less succesful with scanned and photographic images, which may have wide colour ranges and tonal variations.

For these images, *JPEG* is preferred. Strictly speaking, JPEG is a compression technique,[6] and images that have been compressed using it may be stored in any of several file formats — the JPEG standard does not specify any. Colloquially, though, the name 'JPEG file' is used to refer to what are correctly called *JFIF* — JPEG File Interchange Format — files. To complicate matters further, the more recently developed SPIFF format is actually the officially endorsed file format for JPEG images, but as we remarked, JPEG data can be embedded in other files, including TIFF and EPS, which we will say more about shortly.

The third, and newest, file format that is widely supported on the Web is *PNG*,[7] which was devised to supersede GIFs. The trouble with GIF is that the compression algorithm it employs is covered by a patent owned by Unisys, who require a licence fee to be paid for any program that implements GIF compression or decompression.

6 *Really strictly* speaking, JPEG is the Joint Photographic Experts Group, who developed the compression technique, and after whom it is named.

7 Pronounced 'ping'.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 96
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

PNG, on the other hand, uses a different lossless technique that is not encumbered in this way, and can therefore be implemented freely by anybody. Additionally, PNG is not restricted to 256 colours and it offers a more sophisticated form of transparency than does GIF. The PNG format was developed under the aegis of the W^3C, and its specification, published in 1996, has the status of a W^3C Recommendation. Support for PNG has been slow to develop, but it is likely to be supported by all the major Web browsers and graphics programs by the end of 1999.

Outside the World Wide Web, other commonly encountered bitmap graphics file formats include *TIFF*, *BMP*, and *TGA* (often called Targa). TIFF (Tag Image File Format) is an elaborate extensible file format that can store full colour bitmaps using several different compression schemes, including JPEG. It is supported by most painting programs on all platforms, although not all programs are equal in the comprehensiveness of their support, so that TIFF files created by one program cannot always be read by another. TIFF is natively supported by Windows, as is BMP. Indeed, BMP is more properly called the Microsoft Windows Bitmap format. As such, it is platform-dependent, but the near ubiquity of the Windows platforms means that it is widely understood by programs on other systems. Unlike most other bitmapped formats, BMP only supports a simple form of lossless compression, and BMP files are usually stored uncompressed. TGA files have achieved wide currency, because the format was one of the earliest to support more than 256 colours on PCs. It was designed to accompany a proprietary range of video capture boards, but has become more widely supported by programs on most platforms, although its use is probably declining.

All of the formats mentioned so far store bitmapped images. The situation with respect to vector graphics is slightly different. Vector graphics is dominated by PostScript. PostScript, developed by Adobe Systems in the mid-1980s, could be described as a programming language with built-in graphics capabilities that allow it to specify the appearance of pages in the form of a program describing how to place graphic elements — paths, shapes, and fills — on the page. Procedures can be defined, so that an application that generates PostScript (it is not the sort of programming language that anyone would want to write by hand) can create a set of operations suited to its own needs and view of page layout.

➲ Illustrator, for example, when writing PostScript output, prepends a set of definitions which define procedures corresponding to

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 97
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

its drawing primitives, in terms of PostScript's own, lower-level, primitives.

PostScript is intended as a page layout language, which makes it unsuitable for storing single images in isolation, rather than as components of a page. *EPS* (encapsulated PostScript) applies certain conventions to the use of PostScript to ensure that the images in an EPS file are self-contained so that they can be incorporated in other documents. In particular, an EPS file must contain a *bounding box comment*, describing the dimensions of the image. EPS is one of the most widely-used vector graphics formats, but a full PostScript interpreter is required to display EPS images.

In February 1999, the W^{3}C issued a first working draft specification of a Scaleable Vector Graphics format, *SVG*, which is defined in XML (see Chapter 8), the new extensible markup language for the Web. In essence, though, SVG is a derivative of PostScript that uses the same imaging model but a fixed repertoire of operations and is thus easier to implement, and is more compact for transmission over networks.[8] At the same time, the *SWF* format, originally developed for vector animations using Macromedia's Flash (see Chapter 11) but now an open standard, is in wide use for vector images. Although SWF does not have the sanction of the W^{3}C, it has been supported, either via a plug-in or directly, in the major Web browsers for some years,[9] which gives it *ad hoc* status as a standard. SWF is a highly compact format, and can be rendered very quickly. Although, because of its origins, SWF is mostly used for animations, its use for still images is increasing, and may yet pre-empt SVG.

EPS, SWF, and SVG are not just vector formats, although their vector capabilities are their essential feature. It is possible to incorporate bitmaps into these files, too, as self-contained objects. This sort of file format that accommodates both vector and bitmapped graphics, and usually text, too, is sometimes called a *graphics metafile*. Other graphics metafile formats include Macintosh PICT files and Microsoft's Windows Metafiles (WMF). Purely vector-based formats are in fact quite rare and, for the most part, associated with computer-aided design packages: AutoCAD DXF is a complex file format widely used for the exchange of vector data, for example. Vector-based formats are also widely used for three-dimensional models, which we will look at in Chapter 4.

As well as the general purpose formats we have described, certain proprietary formats associated with popular programs are also

8 SVG is closely based on a proposal from Adobe, originally designated PGML (Precision Graphics Markup Language) before the post-HTML vogue for names ending in ML passed.

9 Unlike SVG, which, at the time of writing, has not been implemented at all.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 98
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

widely used. In particular, Photoshop and Illustrator files are commonly used as interchange formats for bitmapped and vector graphics, respectively, in the pre-press and publishing industries. One noteworthy feature of these formats is that they preserve any layers that might have been used in making the image. Exporting, say, a Photoshop image to a PICT file will 'flatten' it, combining all the layers into one. This is undesirable if it is intended to make further changes to the image.

Evidently, with so many different file formats in use, conversions between formats are often required. These vary in complexity from a simple re-ordering of bytes to the sort of vectorization operation described in the previous section. In this context, mention should be made of QuickTime. Although QuickTime is most often considered to be a digital video format (and we will describe it further in Chapter 10) it is better described as a multimedia architecture, which provides a framework and a set of software components for storing and manipulating various digital media formats, of which video is only one — still images are another. QuickTime has its own image format, which generally contains JPEG compressed image data, but more importantly, it provides a collection of import and export components, that allow any program that uses QuickTime to work with files in many formats, including JFIF, PNG, GIF, TGA, PICT, TIFF, Photoshop and SWF (but not EPS).[10]

10 Some of these are only supported by version 4.0 or higher.

Further Information

The topics introduced in this chapter will be taken up in more depth in Chapters 4 to 6. The references given at the end of those chapters should be consulted for further details. [Mv96] describes many graphics file formats; [Ado90b] is the definitive description of the PostScript language.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 99
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Exercises

1. Describe three significant differences between vector and bitmapped graphics.
2. For each of the following kinds of image, which would be more suitable, bitmapped images or vector graphics?
   (a) Circuit diagrams.
   (b) Architectural drawings.
   (c) Botanical drawings.
   (d) Pie charts.
   (e) Fingerprints.
   (f) A map of the world.
   (g) Brain scans.
   (h) Illustrations for a children's alphabet book.
   (i) A reproduction of the Mona Lisa.
   (j) A simple cartoon character, such as Mickey Mouse.
3. Since GIF files are most suitable for the same sort of images as vector graphics, why would you use them in preference to EPS or some other vector format?
4. What file format would you choose to present
   (a) A photograph of yourself.
   (b) A comic strip cartoon drawing.
   (c) The floor plan of a museum.

   for each of the following purposes:
   (a) Incorporation in a Web page.
   (b) Incorporation in a CD-ROM multimedia encyclopedia.
   (c) Transfer to another platform.

   Justify your choice in each case.
5. Do you think the 'layered look' in graphic design is a result of the introduction of layers in graphics programs, a response to the conditions of contemporary life, a reflection of recent ideas in philosophy, or some other cause or combination of causes?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 100
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

4 Vector Graphics

Vector graphics provide an elegant way of constructing digital images whose representation is compact, scaleable, resolution-independent, and easy to edit. The compactness of vector graphics makes them particularly attractive for networked multimedia, where the indiscriminate use of bitmapped images can lead to excessive download times, because of the large sizes of the image files. A common complaint amongst domestic Internet users is that the increasing trend towards graphically rich Web sites means that pages take too long to download. Equally, Web designers who are aware of this complaint feel inhibited about using images freely. Vector images can be a fraction of the size of bitmaps, but the absence of any standard format for vector graphics on the Web left little opportunity for using them. As the official SVG and *de facto* SWF standards are adopted, this will change.

Although vector graphics has been eclipsed in recent years by bitmapped representations for two-dimensional images, for three-dimensional work — that is, images that are constructed as projections of a 3-D model — vector techniques are mandatory, since the use of models made out of the three-dimensional equivalent of pixels (*voxels*) is impractical on all but the most powerful equipment.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 101
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Fundamentals

In vector graphics, images are built up using shapes that can easily be described mathematically. We expect many readers will be familiar with at least the rudiments of coordinate geometry, the field of mathematics underlying the representation of shapes in vector graphics. For the benefit of those who may not be used to thinking about shapes and outlines in terms of coordinates and equations, we will begin with a very brief review of the basic concepts.

Coordinates and Vectors

Since an image is stored as a rectangular array of pixels, a natural way of identifying any single pixel is by giving its column and row number in that rectangular array. If we number the columns from left to right, and the rows from the bottom of the image to the top, both starting at zero, then any pixel is uniquely identified by the pair (x, y), called its *coordinates*, where x is the column number and y the row. In Figure 4.1, the pixel labelled A is at $(3, 7)$, while B is at $(7, 3)$. The point labelled O is at the *origin* $(0, 0)$.

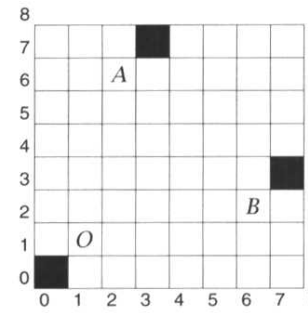


Figure 4.1
Pixel coordinates

The coordinates of pixels in an image must be integer values between zero and the horizontal (for x coordinates) or vertical (for y) dimensions of the image. For the purposes of modelling shapes in a device-independent way, we need to generalize to a coordinate system where coordinates can have any real value. That is, instead of identifying a finite pixel on a grid, coordinates identify infinitely small geometrical points, so that there are infinitely many of them between, for example, $(0, 0)$ and $(1, 0)$. Additionally, the values are not restricted to any finite maximum. We also allow negative coordinates: points with negative x coordinates lie to the left of the origin, while those with negative y coordinates lie below it. Thevertical line running through the origin, which consists of all the points with an x coordinate of zero, is called the *y-axis*, while the horizontal line through the origin is the *x-axis*. We can label the x- and y-axes with the x and y coordinates of equally spaced points to produce the familiar graph axes, as shown in Figure 4.2, from which coordinates can easily be read off. Vector drawing programs usually allow you to display axes (usually called *rulers*) along the edges of your drawing.

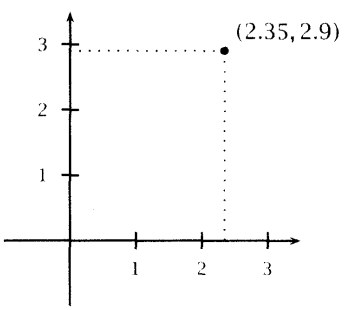


Figure 4.2
Real coordinates and axes

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 102
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

⇨ Although drawing programs (and PostScript) usually follow mathematical convention by using a coordinate system where y values increase upwards, lower level graphics packages, such as QuickDraw or the Java Abstract Windowing Toolkit, often use the opposite convention, with y values increasing downwards. This same convention is employed in some graphics file formats, too. It corresponds more closely to the way pixels are drawn onto physical output devices. You generally should not have to concern yourself with the coordinate system used by the output device. If your drawing program produces PostScript output, then the PostScript interpreter will convert from its coordinate system to that of whichever device your drawing is rendered on.

This conversion is an example of a *coordinate transformation*, whereby coordinates in one system (the *user space*) are transformed into a different one (the *device space*). Coordinate transformations are inevitable if we are to produce device-independent graphics since, in general, we cannot know the coordinate space of the output device. Another example of a coordinate transformation occurs when an image is rendered in a window on a display. Since we cannot know when the image is prepared whereabouts on the screen the window will be positioned, there is no possible way of using absolute screen coordinates to specify the objects in the drawing. Instead, the drawing is prepared in the user coordinate space, and transformed to the device space when it is displayed.

Pairs of coordinates can be used not only to define points, but also to define displacements. For example, to get from A to B in Figure 4.1 we must move 4 units to the right, and 4 units down, or, putting it another way, -4 units up. So we can specify the displacement from A to B by the pair $(4, -4)$. In general, for any pair of points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, the displacement from P_1 to P_2 is $(x_2 - x_1, y_2 - y_1)$, which we write as $P_2 - P_1$ (see Figure 4.3). When a pair of values is used to specify a displacement in this way, we call it a two-dimensional *vector*. Note that a vector has a direction: $P_2 - P_1$ is not the same as $P_1 - P_2$, since moving from P_1 to P_2 is different from moving in the opposite direction from P_2 to P_1.

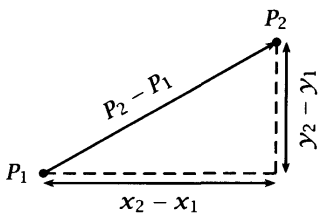


Figure 4.3
A vector

⇨ The term *vector graphics* was originally used to refer to the production of images on output devices where the position of the electron beam of a cathode ray tube was controlled directly by setting its (x, y) coordinates, rather than scanning the screen in a raster pattern. By changing the coordinate values, the beam was made to trace out vectors, hence the name. This type of graphics device only lends itself to the drawing of simple shapes, which is

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 103
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

why the name is retained for the shape-based graphics systems we are describing here.

A coordinate system lets us identify points in space. The power of coordinate geometry comes from using letters to represent 'unknown' values and using equations in those values to specify relationships between coordinates that characterize geometrical shapes. For example, if (x, y) is *any* point on a straight line that passes through the origin at an angle of 45° from south-west to north-east, then it must be the case that $x = y$, as you can show using simple geometry. We can use the methods of coordinate geometry to derive equations for arbitrary straight lines, circles, ellipses, and so on. Using such equations we can represent shapes simply by storing the appropriate constants that appear in the equations, since it is these which distinguish between different shapes belonging to the same class.

➲ Practical considerations might lead us to use a slightly less obvious representation. For example, the equation of a straight line is usually written as $y = mx + c$, where the constants m and c are the slope and intercept, respectively. However, since we can only use finite segments of lines, it is necessary to store the endpoints. The values of m and c can be deduced from these. A bit of simple algebraic manipulation, which many readers will have done at school, demonstrates that, if the line passes through (x_1, y_1) and (x_2, y_2), m is equal to $(y_2 - y_1)/(x_2 - x_1)$ and c is equal to $(x_2 y_1 - x_1 y_2)/(x_2 - x_1)$. Hence, the cordinates of the endpoints are enough on their own to specify both the extent of the line and the constants in its equation.

In a similar way, the values actually stored for other shapes are not necessarily those that a simple mathematical analysis would suggest.

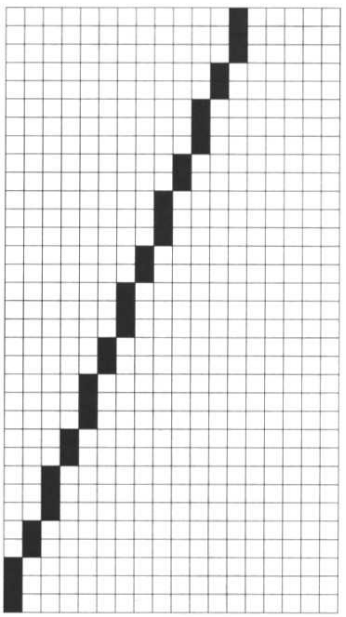


Figure 4.4
Approximating a straight line

When it becomes necessary to render a vector drawing, the stored values are used, in conjunction with the general form of the description of each class of object, to set the values of pixels to form an image of the object described. For example, if a line has endpoints $(0, 1)$ and $(12, 31)$, we could compute the y coordinate corresponding to each integer value as x was stepped from 0 to 12. Remember that a pixel's coordinates are always integers (whole numbers), and we cannot set the value of just part of a pixel. The pixel image can, therefore, only ever approximate the ideal mathematical object which the vector model describes. For example, the line just described has equation $y = 5x/2 + 1$, so

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 104
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

for any odd integer value of x, y must be rounded up (or down — as long as it is done consistently) to get its corresponding integral value. The coordinates of pixels along the line would be $(0,1),(1,4),(2,6),(3,9)\ldots$. To get a continuous line, we set blocks of pixels, but the height of the blocks alternates between 2 and 3 pixels, so that the ideal straight line is approximated by an uneven staircase, as shown in Figure 4.4. This is inevitable, since the output devices we are using are based on a grid of discrete pixels. If the resolution of the output device is low, (i.e. the pixels are relatively large) the jaggedness of lines, and other effects due to the same cause, can become offensive. This phenomenon is colloquially known as 'staircasing' or, more colourfully, 'the jaggies'.

Anti-aliasing

The process of rendering a vector object to produce an image made up of pixels can usefully be considered as a form of sampling and reconstruction. The abstract line that we would like to draw is a continuous signal — the x and y coordinates can vary infinitesimally — which has to be approximated by a sequence of pixel values at fixed finite intervals. Seen in this light, jaggies are a form of aliasing caused by undersampling. This is consistent with the common-sense observation that as the resolution of the output device increases — that is, we sample at a higher rate — the individual pixels get smaller, so that the jagged effect becomes less noticeable.

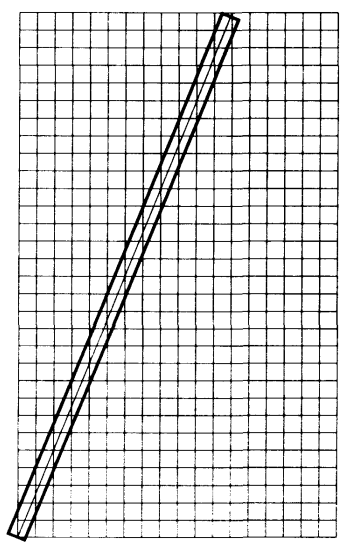


Figure 4.5
The single pixel approximation to a line.

You will recall from Chapter 2 that it is necessary to sample at a rate greater than twice the highest frequency in a signal in order to reconstruct it accurately, and that high frequencies are associated with abrupt changes. If an image contains a sharp hard-edged boundary, its brightness or colour will change directly from one value to another crossing the boundary without any intermediate gradation. The representation in the (spatial) frequency domain of such a discontinuity in the spatial domain will include infinitely high frequencies. Consequently, no sampling rate will be adequate to ensure perfect reconstruction. In other words, jaggies are always possible, no matter how high the resolution that is used for rendering a vector shape.

In any case, practical and financial considerations impose a limit on the available resolution. In particular, vector graphics that are used

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 105
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

in multimedia presentations will usually be rendered on a monitor with a resolution between 72 and 120 dots per inch, and aliasing will be readily visible. To reduce its impact a technique known as *anti-aliasing* is often employed.

Looking back at Figure 4.4, you will see that the staircase effect is a result of the pronounced contrast between black and white pixels. We can soften the effect by using intermediate grey values for some pixels. In terms of the frequency domain representation, we are removing the spurious high frequencies, and replacing them with lower frequencies. We cannot simply tone down the black pixels to produce a grey line instead of a black one; we want to try to use a range of greys to somehow convey to the eye and brain of someone looking at the displayed line the appearance of a smoothness that cannot actually be achieved by the finite pixels.

If we did not have to be concerned about the orientation of the pixel grid, at best we could produce a smooth line one pixel wide, as shown in Figure 4.5. Our original attempt at rendering the line on the basis of its defining equation had the effect of setting to black those pixels whose intersection with this one pixel wide rectangle was at least half the area of the pixel. Anti-aliasing is achieved by colouring each pixel in a shade of grey whose brightness is proportional to the area of the intersection, as shown in Figure 4.6. The number of pixels that are no longer white is greater than before, but if we take the grey pixels and multiply the area of each one by the value used to colour it, the total amount of greyness, as it were, is the same as that produced by only using a single black level to colour the original collection of pixels. At this magnification, the result looks fairly incoherent, although it should be apparent that the jaggedness has been subdued somewhat. At normal viewing resolutions, anti-aliasing can significantly reduce aliasing effects, albeit at the expense of a certain fuzziness.

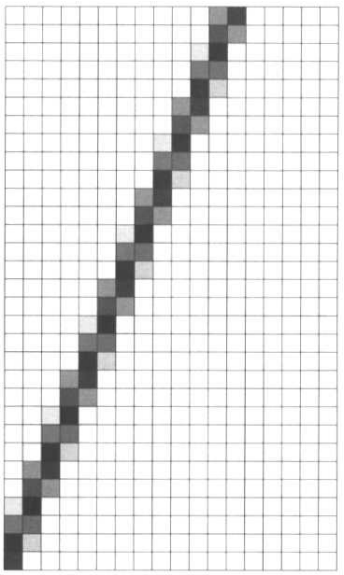


Figure 4.6
Anti-aliased line

Shapes

When using a drawing program you are restricted to the shapes it provides, which are generally shapes with a simple mathematical representation that can be stored compactly and rendered efficiently. Usually the repertoire of shapes is restricted to rectangles and squares (possibly with rounded corners), ellipses and circles,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 106
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

straight lines, polygons, and a class of smooth curves, called *Bézier curves*, although spirals and stars are sometimes supplied too. Shapes built up out of these elements can be filled with colour, patterns or gradients. Because the program works with a description of the shape, not a map of its pixels, it is easy to move, rotate, scale and skew shapes. It may sound as though vector programs are very limited in their graphic capabilities, but they can be used to achieve complex and subtle effects, especially once you understand how to work with Bézier curves. Vector drawing programs must be approached in a different way from a freehand drawing medium, though.

A good way of appreciating the potential and limitations of vector graphics is by looking at the capabilities offered by a typical drawing program. We will use Illustrator as an example; you will find the same concepts and facilities in any other professional drawing package. Illustrator generates PostScript, so the description of Illustrator's facilities provides an indirect account of how graphical objects are constructed by PostScript programs. Generally, the way in which a drawing package allows you to work with shapes is a reflection of the way in which those shapes are represented inside the program or in PostScript code. For example, you draw a line by selecting a pen tool and clicking the mouse or pressure-sensitive pen at each end of the line: internally, a line is represented by the coordinates of its end-points; PostScript's `lineto` operator takes the coordinates of a point and draws a line to it from the current point, established by its `moveto` operator.

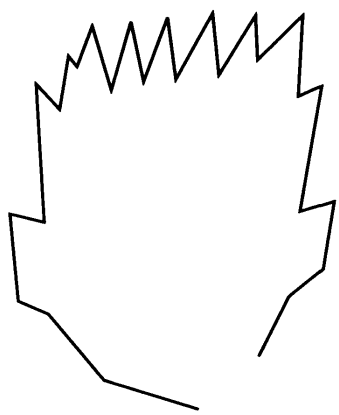


Figure 4.7
A polyline

A sequence of connected lines, such as the one shown in Figure 4.7 is sometimes considered as a single object, called a *polyline*. Closed polylines, whose first and last points coincide, form regular or irregular polygons, and can be used to draw rectangles, for example.

Alternatively, a rectangle whose sides are parallel to the axes can be drawn by selecting the rectangle tool, holding down the mouse button where you want one corner, and dragging to the opposite corner: a rectangle can obviously be completely described by the coordinates of its opposite corners. In Illustrator, rectangles can also be drawn with the centred rectangle tool. With this, you begin at the point where you want the centre to be, and then drag out one corner. It should be clear that it is possible for the program to compute the coordinates of opposite corners from those of the centre and one corner, and *vice versa*.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 107
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Ellipses can be drawn by selecting the appropriate tool and dragging from one point on the perimeter to the opposite point. A pair of points is sufficient to determine the shape and position of the ellipse, and their coordinates can be transformed into one of many convenient representations of the object.

Squares and circles are special cases of rectangles and ellipses, respectively, and so do not need any special representation of their own. It is helpful when drawing to be able to ask the program to restrict rectangles and ellipses to squares and circles. In Illustrator this is done by holding down the shift key while using the rectangle or ellipse tool.

Curves

Lines, polylines, rectangles and ellipses are sufficient for drawing many sorts of technical diagrams (particularly when your lines can be decorated with arrowheads, as they can be in all professional drawing programs). Less constrained drawing and illustration requires more versatile shapes, which are supplied by Bézier curves.

Bézier curves are a class of curve that, as we will shortly demonstrate, have several properties that make them especially useful for graphics. A Bézier curve is completely specified by just four points: its two endpoints, and two more points, called *direction points*, which do not usually lie on the curve itself. The endpoints and direction points are collectively referred to as *control points*. The name 'direction points' indicates the purpose of these points: they show the direction in which the curve sets off from each endpoint. This is shown in Figure 4.8: P_1 and P_4 are the endpoints of the curve, P_2 and P_3 are its direction points, and you can see that the curve is accurately described by saying that it begins at P_1, setting off towards P_2, curving round so that is arrives at P_4 from the direction of P_3.

The length of the lines from each endpoint to its direction point determine how wide a sweep the curve makes. You can think of the lengths of these lines as representing the speed with which the curve sets off towards the direction point: the faster it goes, the further out it will curve.

This characterization of the curve is the basis for the way Bézier curves are drawn interactively. After selecting the appropriate tool (usually a pen), you click at the first endpoint, and then drag out

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 108
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

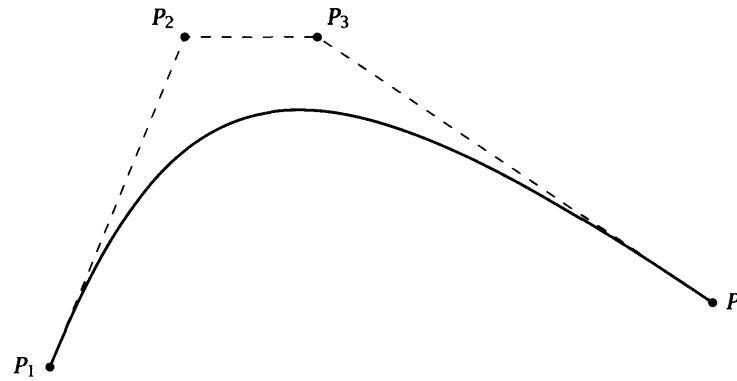


Figure 4.8
A Bézier curve

towards the first control point, as if you were pulling the curve towards it. You will usually see a *direction line* showing you how far you have pulled. In most applications, for reasons that will be explained in the next section, the direction line usually extends away from the endpoint both in the direction you drag and symmetrically in the opposite direction. Once you have the first one right, you click at the point where you want the curve to end, and drag away from the direction point (see Figure 4.9). You will see the curve being formed as you move the cursor. If you do not like the result when you have finished, you can subsequently select any of the control points and drag it around to change the shape and extent of your curve.

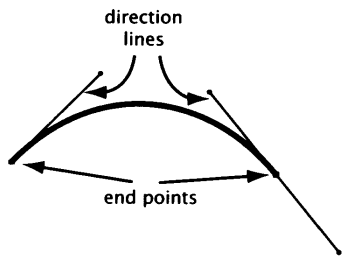


Figure 4.9
Drawing a curve with the pen tool in Illustrator

⇨ It is perhaps not immediately obvious that a curve related to four points in the manner just described is unique, or even that it always exists. Since this is not a book about the mathematics of Bézier curves, you will have to take it on trust that it is and does, or consult one of the references at the end of the chapter. However, it may help to contemplate the following construction for building curves from a pair of endpoints P_1 and P_4 and a pair of direction points P_2 and P_3.

Begin by finding the mid-points of the lines between P_1 and P_2, P_2 and P_3, and P_3 and P_4. Call these P_{12}, P_{23} and P_{34}, respectively, and construct the lines between P_{12} and P_{23} and between P_{23} and P_{34} (see Figure 4.10). Next, find the mid-points P_{123} and P_{234} of these new lines, join *them* together (Figure 4.11) and find the mid-point of this last line. This final mid-point, Q lies on the curve. (Figure 4.12.)

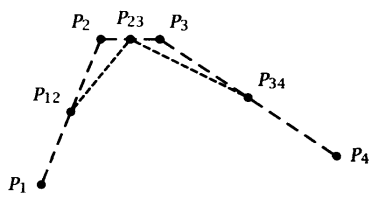


Figure 4.10
Constructing a Bézier curve, step 1

Q lies on the curve, because we assert that this construction is how our curve is to be built. We proceed by taking the two sets of four

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 109
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

points P_1, P_{12}, P_{123} and Q, and Q, P_{234}, P_{34} and P_4, and asserting that the curves built using them as control points will be the left and right halves of our complete curve. We therefore apply the same technique of finding mid-points and so on to produce two points on the portions of the curve to the left and right of Q. This in turn leaves us with new sets of control points from which to construct the portions of the curve lying to each side of the new points we have found.

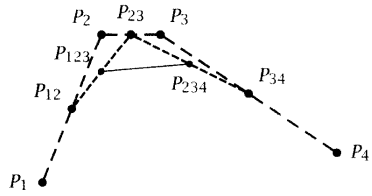


Figure 4.11
Constructing a Bézier curve, step 2

If we were mathematicians, we would go on bisecting our curve forever, until the curve segments became infinitesimally small. If you actually try this construction on paper, or write a program to perform it for you, you will find that the segments very rapidly become indistinguishable from straight lines and your curve becomes as smooth as your drawing medium or display will allow. Figure 4.13 shows how the left half of the curve produced by the construction fits between its control points the same way the complete curve fits between P_1, P_2, P_3 and P_4.

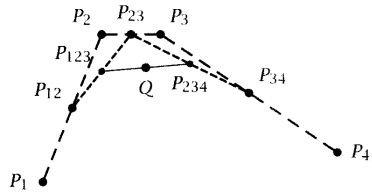


Figure 4.12
Constructing a Bézier curve, step 3

However much bisecting we do, the construction lines that pass through the endpoints P_1 and P_4 are part of the lines between P_1 and P_2 and between P_3 and P_4. This remains the case, even if we were to go on bisecting to infinity, so that, in the limit, these lines would be the tangents to the curves at P_1 and P_4. Which is to say that the curve leaves P_1 heading towards P_2, and approaches P_4 from the direction of P_3, as originally described. Furthermore, the further away P_2 is from P_1 the further away will be the midpoint, and, eventually, the curve points we construct, so that the length of the tangents controls how widely the curve sweeps round.

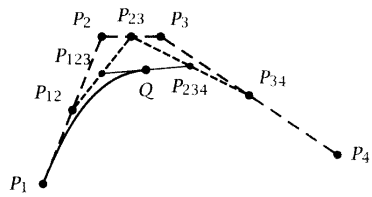


Figure 4.13
The left half of the curve

Mathematicians will find the preceding account a bit glib, and may feel more comfortable with an alternative description of Bézier curves.

The curves used in most drawing programs are, strictly speaking, Bézier *cubic* curves, a form of cubic with certain desirable properties. They are usually defined by the *parametric equations*:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + x_1$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + y_1$$

where the endpoint $P_1 = (x_1, y_1)$. The other control points are then given by:

$$P_2 = (x_2, y_2) = (x_1 + c_x/3, y_1 + c_x/3)$$
$$P_3 = (x_3, y_3) = (x_2 + (c_x + b_x)/3, y_2 + (c_y + b_y)/3)$$
$$P_4 = (x_4, y_4) = (x_1 + c_x + b_x + a_x, y_1 + c_y + b_y + a_y)$$

The curve defined by these four control points is the path traced out by $x(t)$ and $y(t)$ as t varies from 0 to 1. We invite you to show that

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 110
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

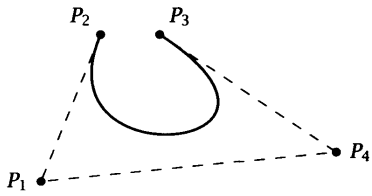


Figure 4.14
P_2, P_1, P_4, P_3

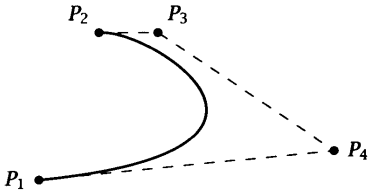


Figure 4.15
P_1, P_4, P_3, P_2

the lines P_1P_2 and P_3P_4 are the tangent vectors to the curve at its endpoints.

There is an interesting physical interpretation of this parametric formulation of the Bézier curve: if t denotes time, then $x(t)$ is a function showing how x, which could denote horizontal displacement, varies over time, while $y(t)$ describes how vertical displacement varies with time. The two equations together thus describe a two-dimensional trajectory, traced by a point over time from $t = 0$ to $t = 1$. The Bézier curve can be extended into three dimensions. In that case, the equations might describe the trajectory of, for example, a camera through a three-dimensional space. As we will see in Chapter 11, such trajectories are commonly used in three dimensional computer-generated animation.

You can construct a Bézier curve using any set of four control points, but the result is not necessarily going to be useful or lovely. Nor is it always obvious, until you have acquired some experience using these curves, exactly what the curve built from any four control points is going to look like. Figures 4.14 to 4.18 show the curves produced from the same set of points as were used in Figure 4.8, but in different orders.

Paths

A single Bézier curve on its own is rarely something you want in a drawing. Except in stylized applications for which rectangles, ellipses and straight lines serve adequately, we need to be able to draw a variety of irregular curves and shapes, such as those making up the petals and stamens of the poppy in Figure 3.2 on page 71. In principle, because the pixels on monitors and printers are finite in size, any shape, no matter how curvaceous, can be approximated as well by a collection of straight lines as by any other method. However, in practice, to produce acceptable approximations to curved shapes, we need to use a lot of very short lines. We thus lose much of the advantage of vector graphics, inasmuch as our descriptions of shapes become large, unwieldy and difficult to work with and edit interactively.

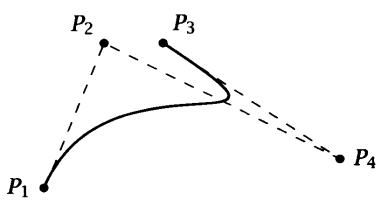


Figure 4.16
P_1, P_2, P_4, P_3

What makes Bézier curves useful is the ease with which they can be combined to make more elaborate curves and irregular shapes. Remember that a Bézier curve with control points P_1, P_2, P_3 and P_4 approaches its endpoint P_4 from the direction of P_3. If we construct a second curve with control points P_4, P_5, P_6 and P_7 (so that it is

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 111
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

joined to the original curve at P_4), it will set off from P_4 in the direction of P_5. Provided we ensure that P_3, P_4 and P_5 are in a straight line, with P_5 on the opposite side of P_4 to P_3, the curve segments will both be travelling in the same direction through P_4, so there will be a smooth join between them, as shown in Figure 4.19. The join in Figure 4.20 is smoother still, because we have made sure that the length of the direction lines is the same on each side of P_4. If you think of the line as a trajectory through space, this ensures that it passes through the shared endpoint at a constant velocity, whereas, if we only make sure the three points are in a straight line, the direction is constant, but the speed changes.

The smoothness of joins when control points line up and direction lines are the same length is the reason behind the display of direction lines in drawing programs. When you drag towards an endpoint, as we saw earlier, direction lines are displayed both to and from it. In other words, you are shown two direction points: one belonging to the curve you are just finishing, the other which could belong to a new curve which joins smoothly on to the end of it. Thus, you can rapidly build up compound curves using a sequence of dragging movements at the points where you want Bézier segments to join.

⇨ This property of continuity is not shared by other sorts of curve which might seem to be candidates for the job of curve-drawing primitive. You cannot generally make arcs of circles, parabolas or ellipses join smoothly.

Sometimes you will want your curve to change direction instead of continuing smoothly. To do so, you simply need to arrange that the direction lines of adjacent segments are not lined up (see Figure 4.21). In a drawing program, some expedient is required to break out of the default behaviour of smoothly joining curves. In Illustrator, for example, you hold down the option (or alt) key after creating the direction lines at a point; you are then able to drag the direction point for the new curve segment round to where you want it, to make an abrupt corner. By mixing clicking and dragging, it is possible to combine curves and straight lines in arbitrary ways.

Some additional terminology is used to talk about joined curves and lines. A collection of lines and curves is called a *path*. If a path joins up on itself, it is said to be *closed*, otherwise it is *open*. Figure 4.23 shows a closed path; Figure 4.22 shows an open one. Open paths have endpoints, closed paths do not. Each individual

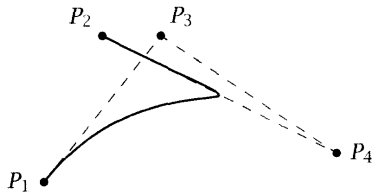


Figure 4.17
P_1, P_3, P_4, P_2

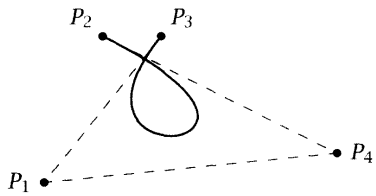


Figure 4.18
P_3, P_1, P_4, P_2

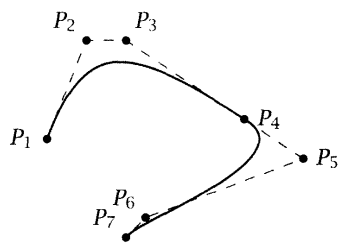


Figure 4.19
Joining two Bézier curves

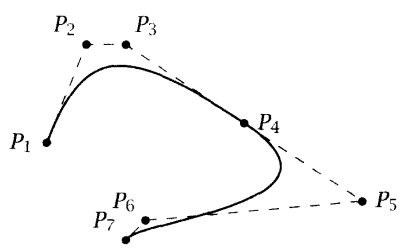


Figure 4.20
A smoother join

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 112
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

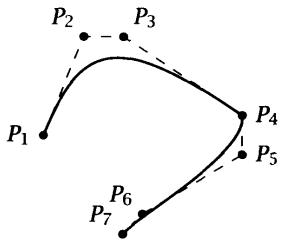


Figure 4.21
A corner point

line or curve is called a *segment* of the path; the points where segments join (the original endpoints of the segments) are called the path's *anchor points.* Note that any collection of curves and lines may be considered as a path, they do not all have to be connected.

The usual way to construct a path is by constructing the individual segments with the pen tool, which provides great control and permits you to construct very accurate curves. However, if you wish to create artwork with a hand-drawn look, you can use Illustrator's pencil tool. With this tool selected, you can just drag with the mouse or pressure-sensitive pen, as if you were drawing freehand with a pencil or pen. You are not doing so: Bézier curve segments and straight lines are being created to approximate the path your cursor follows. Once the path is complete, you can select it and see anchor points that you can adjust in the usual way. The faithfulness of the approximation can be controlled by a tolerance setting. A higher tolerance leads to a more efficient path, with fewer anchor points, which may, however, smooth out some of the smaller movements you made with the pencil tool.

Stroke and Fill

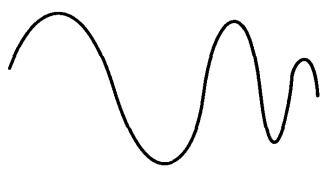


Figure 4.22
An open path

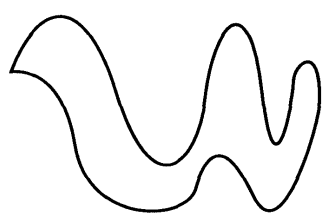


Figure 4.23
A closed path

A path, strictly speaking, is an abstract mathematical entity: just as points are infinitesimally small, so a path is infinitesimally thin. Although we have talked about drawing curves, and shown you pictures of paths, you cannot really see a path. You can, however, use it as a specification of something you *can* see. You can do this in two different ways. Either you apply a *stroke* to the path, making it visible as if you had traced it with ink or some other medium, or you treat the path as the outline of a shape, and *fill* it, as if with ink or paint. Or both. Since computer graphics is not bounded by the physical limitations of real art materials, you can stroke or fill paths with more elaborate things than flat colour, such as patterns or gradients.

⇨ Practically, you have to be able to see your path while you are drawing it, of course, so Illustrator shows you the path as a thin stroke. Once it is complete, each path segment is stroked or filled straight away, so you can see what you are doing.

Consider first applying a stroke to a path. Like physical strokes on paper, the strokes applied by a drawing program have characteristics, such as weight and colour, which determine their appearance.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 113
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

These characteristics can be set and changed by the user of a drawing program. The weight is usually set by specifying the width of strokes numerically, in whatever units are convenient. Specification of colours is more complex, and is described in Chapter 6.

⇨ As we noted in Chapter 3, some drawing programs can apply strokes that simulate natural media, such as charcoal or painted brush-strokes.

It is customary for drawing programs to support dashed strokes as well as solid ones. Ideally, the length of dashes and of the gaps between them can be specified. Again, this is usually done by the user entering numerical values in appropriate units.

A more subtle feature of strokes is the shape of their ends — the *line cap*. If a stroke has any appreciable thickness, cutting it off square at the ends with a *butt cap* may produce an undesirable and ugly effect. It may be preferable to use a *round cap*, where the line is finished off with a filled-in semicircle built across its end. A third option is to use a *projecting cap*, with the stroke continued beyond the endpoint of the path by half the width, so that the weight of the stroke relative to the path is the same in all directions. These three line cap options are provided by PostScript, and are consequently supported by drawing programs like Illustrator that produce PostScript output. Combining different line caps with dash patterns provides a range of effects, as shown in Figure 4.24.

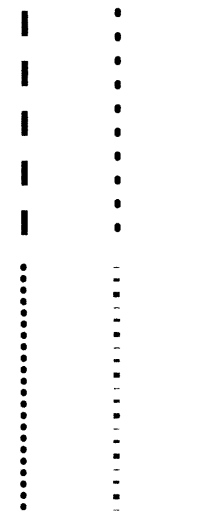


Figure 4.24
Dashed effects

Joins at corner points also need consideration, because wide lines can only meet cleanly if they do so at 90°; when they meet at any other angle an unsightly gap or overlap will result. This can be removed in several ways. The three styles of line join provided by PostScript are a *mitre* — as in a picture frame, the outside edges of the lines are extended to meet at a point; *round* — a circular arc is used to produce a rounded corner; and *bevel* — the segments are finished off square where they join, and the resulting notch is filled in with a triangle to produce a flat ended joint. If mitred joins are used on segments that meet at a very narrow angle, long projecting spikes will result. To avoid this, a limit can be specified, and if the ratio of the spike's length to the stroke width exceeds it, mitres will be replaced by bevels. Figure 4.25 illustrates the different joining styles.

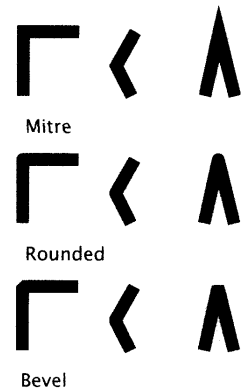


Figure 4.25
Joining styles

As well as applying a stroke to a path, you can use it as an outline and fill it. Astute readers will observe that you can only fill a closed

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 114
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

path, but most drawing programs also allow you to fill an open path — the filling operation implicitly closes the path with a straight line between its endpoints.

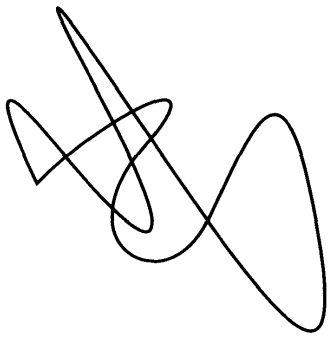


Figure 4.26
A complex path...

The simplest fill is a single colour, and this is often all that is required. When a fill (or, indeed, a stroke) is applied it completely obscures anything underneath it. There is no mixing of overlaid colours, as you would expect if you were using watercolour paints, for example. This means that, among other possibilities, you can use a shape filled with the background colour to knock out areas of objects underneath it.

More interesting and sometimes more attractive effects can be produced by using *gradient fills* and *patterns*.

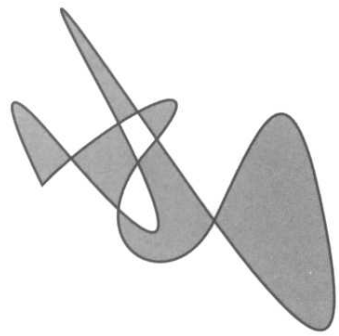


Figure 4.27
...filled

Plate 5 shows two examples of gradient fills. This type of fill is characterized by a gradual transition between colours or tones. In the simplest case — a *linear* gradient — the colours at each end of a region are specified, and a smooth blend of intermediate colours is generated in between. Illustrator provides controls to let you specify intermediate colours, adjust the mid-point of the gradient (where the two colours being blended are of equal intensity) and the line along which the gradient varies. An alternative form of blending, also shown in Plate 5, has the colour varying outwards from a centre point to the outside of the fill. This is called a *radial* gradient. The more sophisticated gradients used in the Plate 6 were created using Illustrator's gradient mesh tool, which allows the designer or artist to set up a two-dimensional mesh of points, and specify the colours at each mesh point; the colours are blended in the spaces between the mesh points. The shape of the mesh can be adjusted dynamically until the desired effect is achieved.

Gradient fills are very widely used in artwork created in vector drawing programs, and contribute to the characteristic air-brushed look of much of the graphic design that is produced using these programs.

Pattern fills, as the name suggests, allow you to use a repeating pattern to fill in an area, as shown in Plate 7. Patterns are built out of elements called *tiles*. A tile is just a small piece of artwork, made using the facilities provided by your drawing program (possibly including the facility to import bitmaps as objects). The name embodies the analogy normally employed for describing how an area is filled with a pattern. Imagine that the artwork is rendered onto a rectangular ceramic tile, such as you might use in your

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 115
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

bathroom. Copies of the tile are arranged in rows and columns parallel to the x- and y-axes, butted together as if by a skilled tiler. The resulting pattern is clipped to the area being filled. Constructing tiles that join seamlessly requires a certain amount of skill — in graphics as in bathroom design. Tiles may be used to produce geometrically patterned areas, such as might be appropriate for drawing textiles or wallpaper (or bathrooms), but they can also be designed so that they produce textured effects. Pattern fills are often used as backgrounds.

Some drawing programs allow you to use patterns to stroke paths, producing a textured outline, for example. This is more difficult than using patterns as fills, because one naturally wants the tiles to be arranged perpendicular to the path, not horizontally. This in turn makes it difficult to get tiles to go round corners, so that a pattern intended for tiling a path must include special corner tiles.

If you want to fill a path, you need to know which areas are inside it. For simple shapes, this is a trivial question, but a path may be arbitrarily complex, crossing itself several times. Figure 4.26 shows a single path. Which areas are inside it, and which outside? There is no absolute answer; several different interpretations of the concept of insideness have equal validity. Figure 4.27 shows Illustrator's answer, which is based on the *non-zero winding number rule*, which may be expressed as an algorithm as follows: To determine whether a point is inside a path, draw a (conceptually infinite) line from the point in any direction. Start by setting the winding number to zero. Follow the constructed line, and every time the path crosses it from left to right, add one to the winding number; every time the path crosses from right to left, subtract one from the winding number. After all crossings have been counted, if the winding number is zero, the point is outside the path, otherwise it is inside. Note that this algorithm depends on the path's having a direction, which will depend on the order in which anchor points were added to it.

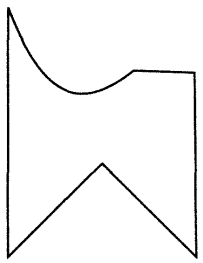


Figure 4.28
A simple object...

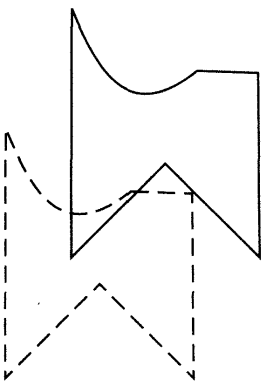


Figure 4.29
...translated...

Transformations and Filters

The objects that make up a vector image are stored in the form of a few values that are sufficient to describe them accurately: a line by its endpoints, a rectangle by its corners, and so on. The actual pixel values that make up the image need not be computed until it is

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 116
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

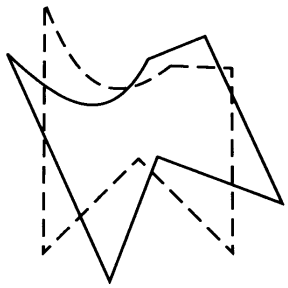


Figure 4.30
...rotated...

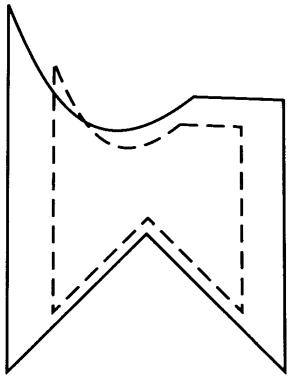


Figure 4.31
...scaled...

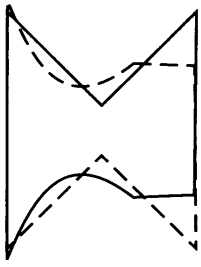


Figure 4.32
...reflected...

displayed. It is easy to manipulate objects by changing these stored values. For example, if a line runs parallel to the x-axis from $(4,2)$ to $(10,2)$, all we need do to move it up by 5 units is add 5 to the y-coordinates of its endpoints, giving $(4,7)$ and $(10,7)$, the endpoints of a line running parallel to the x-axis, but higher up. To use a term we introduced earlier in this chapter, we have transformed the image by editing the *model* that is stored in the computer.

Only certain transformations can be naturally produced in this way. The most important are *translation* — a linear movement of the object — *scaling*, *rotation* about a point, *reflection* about a line, and *shearing* — a distortion of the angles of the axes of an object. These transformations are illustrated in Figures 4.28 to 4.33. Any modern drawing program will allow you to perform these transformations by direct manipulation of objects on the screen. For example, you would translate an object simply by dragging it to its new position.

Briefly, the operations on the model which achieve these transformations are as follows. Any translation can be done by adding a displacement to each of the x and y coordinates stored in the model of the object. That is, to move an object Δ_x to the right and Δ_y upwards, change each stored point (x,y) to $(x+\Delta_x, y+\Delta_y)$. Negative Δs move in the opposite direction. Scaling is performed by multiplying coordinates by appropriate values. Different factors may be used to scale in the x and y directions: to increase lengths in the x direction by a factor of s_x and in the y direction by s_y, (x,y) must be changed to $(s_x x, s_y y)$. (Values of s less than one cause the object to shrink.) Thus, to double the size of an object, its stored coordinates must be multiplied by two. However, this has the effect of simultaneously displacing the object. (For example, if a unit square has its corners at $(1,2)$ and $(2,1)$, multiplying by two moves them to $(2,4)$ and $(4,2)$, which are the corners of a square whose side is of length 2, but it is now in the wrong place.) To scale an object in place, the multiplication must be followed by a suitable, easily computed, displacement to restore it to its original position.

Rotation about the origin and reflection about an axis are simple to achieve. To rotate a point (x,y) around the origin in a clockwise direction by an angle θ, you transform it to the point $(x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$ (which you can prove by simple trigonometry if you wish). To reflect it in the x-axis, simply move it to $(x,-y)$; in the y-axis, to $(-x,y)$. Applying these operations to all the points of an object will transform the entire object. The more general operations of rotation about an arbitrary point and reflection in an arbitrary line require more complex, but

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 117
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

conceptually simple, transformations. The details are left as an exercise.

Finally, when an object is sheared, it is as if we took the x-axis and skewed it upwards, through an angle α, say, and skewed the y-axis through an angle β (see Figure 4.34). You can show that the transformation can be achieved by moving (x, y) to $(x + y \tan \beta, y + x \tan \alpha)$.

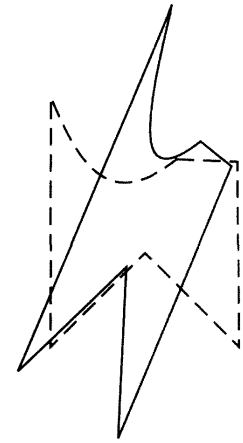


Figure 4.33
... sheared

Other, less structured, transformations can be achieved by moving (i.e., changing the coordinates of) the anchor points and control points of paths. This would normally be done by interactive manipulation in a drawing program. Anchor points and control points may also be added and deleted from paths, so that a designer or artist can fine-tune the shape of objects.

Some commonly required effects which fall between the highly structured transformations and the free manipulation of control points are provided as *filters* in Illustrator and similar programs. (The term 'filter' is taken from photography, where optical filters are used to produce similar effects.) An object is selected, and a filter operation is chosen from a menu of those available. The chosen effect is then applied to the selected object. Filters available in Illustrator include roughening, which produces a rough edge to an object by moving the anchor points of its path in a jagged pattern, scribbling, which moves the anchor points in a random fashion, and rounding corners, which converts corner points into smooth curves. Most filters are parameterized in values such as the maximum distance an anchor point may be moved. The parameters can be set by the user via controls such as sliders.

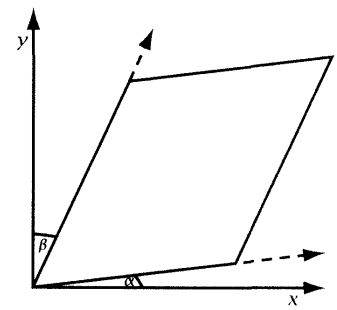


Figure 4.34
Skewed axes

The important thing to understand about all these transformations is that they are achieved simply by altering the coordinates of the defining points of objects, altering the stored model using nothing but arithmetical operations which can be performed efficiently. Although every pixel of the object must be transformed in the final displayed image, only the relatively few points that are needed to define the object within the model need to be re-computed beforehand. All the pixels will appear in the desired place when the changed model is rendered on the basis of these changed values.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 118
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

3-D Graphics

Pictures on a screen are always two-dimensional, but that doesn't mean that the models from which they are generated need to be restricted to flat two-dimensional shapes. Models of three-dimensional objects correspond more closely to the way we perceive space. They enable us to generate two-dimensional pictures as perspective projections — or perhaps other sorts of projection — onto a plane, as if we were able to photograph the model. Sometimes, this may be easier than constructing the two-dimensional image from scratch, particularly if we can begin with a numerical description of an object's dimensions, as we might if we were designing some mechanical component, for example, or constructing a visualization on the basis of a simulation. A three-dimensional model allows us to generate many different images of the same objects. For example, if we have a model of a house, we can produce a view of it from the front, from the back, from each side, from close up, far away, overhead, and so on, all using the same model. If we were working in only two dimensions, each of these images would have to be drawn separately. Working in three dimensions really begins to open up new creative possibilities when we start to generate a sequence of images as we move models and cameras in space, as we will see in Chapter 11.

3-D graphics, as we call vector graphics based on three-dimensional models, is a complicated subject, though, requiring tools that are hard to master, and should be left to specialists most of the time. Here, we can only outline the main features of 3-D technology, in order to provide an appreciation of the difficulties it presents and the opportunities it has to offer.

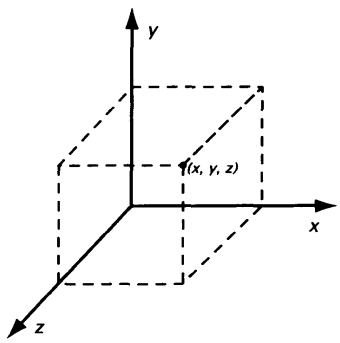


Figure 4.35
Axes in three dimensions

In abstract mathematical terms, generalizing coordinate geometry from two dimensions to three is straightforward. The x- and y-axes of a two-dimensional coordinate system are perpendicular to each other. If you imagine drawing a set of axes on a flat sheet of paper and pinning it to a vertical wall, you can see that we can place a third axis perpendicular to the other two, coming out of the paper horizontally. Just as we can use x- and y coordinates to define a point's horizontal and vertical distance along the wall from the origin, we can use a z-coordinate, measured along our third axis, to define its distance from the wall. The three coordinates together define a point in a three-dimensional space (see Figure 4.35).

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 119
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

The primitive geometrical shapes of 2-D graphics are replaced by 3-D objects: instead of a circle, we have a sphere, instead of a square, a cube, and so on. By an obvious extension, a three-dimensional vector defines a displacement in the same way the two-dimensional vectors we used earlier did. This allows us to describe translations of objects in three-dimensional space. We certainly have a basis for 3-D vector graphics.

Even at this stage, it is apparent that three dimensions may be more than one and a half times as complicated as two. Consider rotation. In two dimensions we rotate about a point; in three, we must rotate about a line. Rotations about an arbitrary line can be built out of rotations about the axes, but that still leaves three distinct rotations to consider, as shown in Figure 4.36, which introduces the names used for each of them.

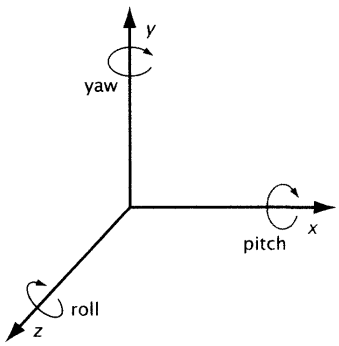


Figure 4.36
Rotations in three dimensions

⇨ When we introduced a third dimension, we stated that the z-axis points out of the imaginary wall of the x-y plane. Why should it not point into it? There is no reason. The coordinate system we have adopted is no more than a convention. It is known as a *right-handed* coordinate system, a name that Figure 4.37 should explain. A left-handed system is equally valid, and is sometimes used in computer graphics. However, the right-handed system is more widely used, especially by mathematicians, and we will employ it exclusively.

We have also adopted the convention that the vertical axis is labelled y, as it was before, but some systems use the x- and y-axes to define a horizontal *ground plane*, with the z-axis adding height to it. In that case, the names of the three rotations are assigned differently to the three variables, although they retain the same spatial meaning — roll is always a movement around the front-to-back axis, and so on.

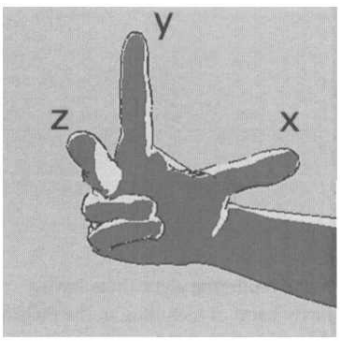


Figure 4.37
Right-handed coordinate system

The added complexity of graphics employing the third dimension goes much further than an extra couple of rotations. Instead of defining shapes by paths, we must define objects by surfaces, which require more complicated mathematics and are much harder for most people to visualize. The fact that we can only work with two-dimensional representations while building models makes visualization generally difficult. This problem is further exacerbated by the fact that only very powerful workstations can provide real-time rendering of 3-D models, so it is usually necessary to work with cruder approximations during the design process.

Once a collection of objects has been modelled, they are arranged in space, usually interactively. Often, a complete scene will be

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 120
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

constructed from a few objects that are specially modelled and others taken from a library, either developed in earlier projects or bought in as the 3-D equivalent of clip art. (Indeed, some 3-D applications intended for consumer or corporate use allow you to do little more than arrange ready-built models in space.)

As well as the spatial relationships between separate objects, we may also wish to consider the relationships between objects that form parts of a larger object. Most complex objects have a hierarchical structure. They can be described as a collection of sub-objects, each of which might be a collection of sub-sub-objects, and so on, until we reach the smallest component objects, which can be modelled in a simple way. For example, a bicycle consists of a frame, two wheels, a saddle, handlebars and front fork. Each wheel is a tyre around a rim, with spokes and a hub. We could model the hub as a cylinder, without decomposing it further. Hierarchical modelling is a well-established way of coping with complexity in many fields. In 3-D it becomes of particular importance when objects are animated, because we can then take advantage of the relationships between components to deduce how the object must move as a whole.

Rendering is no longer a relatively simple matter of generating pixels from a mathematical description. In 3-D, we have a mathematical model of objects in space, but we need a flat picture. Assuming that we want to use conventional Renaissance perspective to project the 3-D model onto a plane surface, we will need to consider the viewpoint, or the position of an imaginary camera, and ensure that the rendered picture corresponds to what the camera sees,[1] while exhibiting the scaling with distance that we expect from perspective.

1
Naïve rendering algorithms have a nasty habit of including in the picture things that are behind the camera or on the hidden side of objects.

We also need to consider lighting — the position of light sources, and the intensity and type of illumination they cast, whether it is a diffuse glow or a concentrated beam, for example. The interaction of the light with the surface of objects, and perhaps — in the case of an underwater scene or a smoke-filled room — with the atmosphere must also be modelled and rendered. If we want to achieve any sort of realism, our models must include not just the geometrical features of objects, but also their surface characteristics — not just the colours, but also the texture — so that the surface's appearance can be rendered convincingly in different positions and under different lighting conditions. Although 3-D systems are reasonably good at using lighting models derived from the underlying physics, they are not perfect, and designers sometimes have to resort to

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 121
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

physical impossibilities, such as negative spotlights, for absorbing unwanted light.

These added complications mean that the theory of 3-D graphics is a great deal more elaborate than that of 2-D graphics. It also means that 3-D software is more complex and difficult to use. Finally, it means that rendering 3-D models can be an extremely computationally expensive process, that often requires additional hardware (often in the form of 3-D accelerator PCI cards) to achieve acceptable performance on desktop machines.

3-D Models

Broadly speaking, there are three general approaches to modelling objects, which are often used in conjunction. The simplest approach, which goes by the name of *constructive solid geometry*, uses a few primitive geometric solids, such as the cube, cylinder, sphere and pyramid, as elements from which to construct more complex objects. These elements can be distorted, by squashing or stretching, to produce variations on the simple forms. They can also be combined, using operations usually described in terms of the set theoretical operators union, intersection, and difference.

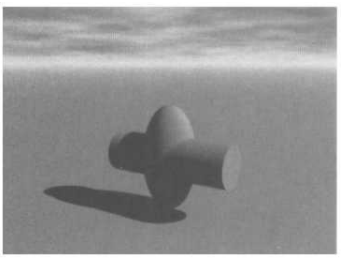


Figure 4.38
Union

These operations only do anything to two objects that are placed in such a way that they share some of the space that they occupy — normally a physical impossibility, but no problem for a computer model. Their *union* is a new object made out of the space occupied by the two together. Figure 4.38 shows the object formed from the union of a horizontal cylinder and a vertical ellipsoid.[2] The *intersection* of two objects is the space that the two have in common; Figure 4.39 shows what happens when the same cylinder and ellipsoid are intersected: the shape that is produced is otherwise difficult to describe. Finally, the *difference* of two objects is the space occupied by the first but not the second. This operation is useful for knocking holes out of solid objects, as Figure 4.40 shows.

2
The shadow and background are just cosmetic additions.

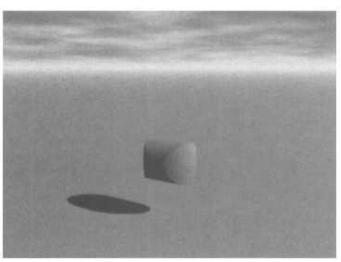


Figure 4.39
Intersection

Constructive solid geometry is especially useful for modelling man-made objects and architectural features, which are often built out of the same elementary solids. It is often found in computer-aided design systems. It can take you a long way with modelling camshafts, triumphal arches and toy steam trains. However, as a glance around your immediate environment will show, many objects

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 122
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

in the real world are not made out of geometric solids combined by set operations; these need a different approach.

Free form modelling uses a representation of an object's boundary surface as the basis of its model. This approach is a 3-D generalization of the use of paths to enclose shapes in two dimensions. Instead of building paths out of lines and curves, we must build surfaces out of flat polygons or curved patches. Surfaces constructed as a *mesh* of polygons can be rendered relatively efficiently, making this a popular representation.[3] However, they suffer from the drawback that, like straight line segments used to approximate a curve, polygons cannot fit together smoothly when they are used to approximate curved surfaces. This is more serious than it may sound, because it affects the way light is reflected off the surface. If reflections are broken up, any irregularities will be readily apparent.

3
Where fast rendering is required, as in 3-D games, the polygons are often restricted to triangles, which makes rendering more efficient. It also guarantees that the polygons are flat.

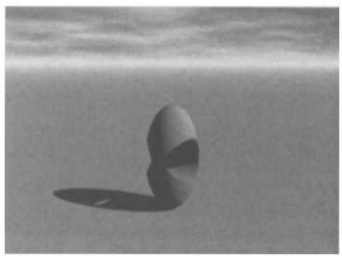


Figure 4.40
Difference

➲ It is possible to generalize Bézier curves to three-dimensional surfaces in order to produce curved patches that can be fitted together smoothly. A cubic Bézier surface patch requires sixteen control points, instead of the corresponding curve's four. Just as we could join curves together smoothly by ensuring that they meet at an anchor point and the connected control points are in a straight line, so we can join patches by ensuring that they meet at a common edge curve and that the connected control points lie in the same plane. This is easy to say, but Bézier patches are hard to work with in interactive 3-D applications, largely because moving a single control point can affect the geometry of the whole patch in a way that may be hard to predict. A more tractable kind of patch is based on surfaces called *non-rational B-splines* or *NURBs*, which, by using a more complicated construction, ensure that the effect of moving control points is localized. In effect, this makes it possible to sculpt a smooth surface by pulling and pushing it into the desired shape. NURBs are largely confined to high-end systems.

The generality and lack of structure of boundary representations can make it hard to get started with a model. To overcome this problem, most 3-D programs provide a means of generating objects with a certain regular structure or symmetry from 2-D shapes. The resulting objects can then either be used directly, or their surface mesh can be altered to produce a less constrained object.

The basic idea is to treat a two-dimensional shape as a cross section, and to define a volume by sweeping the cross section along a path. The simplest path is a straight line. A shape creates an object

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 123
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

with a uniform cross section as it travels along a straight line. For example, a circle creates a cylinder in this way. This process is known as *extrusion*, since the objects it produces resemble those that can be made by industrial processes in which plastic or metal is forced through an opening. Extruded text is an application of this technique that has been so widely used in producing corporate logos as to have become a cliché. To produce more elaborate objects, a curved path can be used, and the size of the cross section can be altered as it moves along it. If the path is a conventional Bézier path, organic shapes can be generated. If it is a circle, the resulting objects exhibit radial symmetry. If a suitable shape is chosen, circular paths can be used to generate many types of drinking vessel and vase, as well as mechanical components. Because of the resemblance of the resulting objects to traditional turned artefacts, this special case is often called *lathing*. Figure 4.41 shows some simple objects constructed by extrusion and lathing.[4]

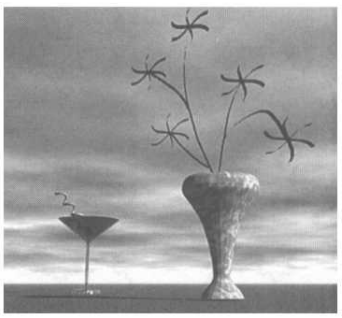


Figure 4.41
Lathed and extruded objects

4
The flowers are slightly extruded from a drawing you can find in Chapter 13. This is enough to make them into objects that can be positioned in three dimensions.

The third appraoch to modelling is *procedural modelling*. Here, instead of using models that can be described by equations, and storing only the constants that define a particular instance, we use objects that are described by an algorithm or procedure. Thus, returning to two dimensions for a moment, instead of defining a circle by the equation $x^2 + y^2 = r^2$, we could define it by some procedure such as 'draw a curve that maintains a constant distance r from the origin'. In this case, the procedural representation is not very helpful — the equation tells you how to implement the procedure — but for naturally occurring objects with a more complex, less mathematically tractable, structure, algorithms may provide a description where equations cannot.

The best known procedural modelling techniques are based on *fractals*. These are often described as shapes that exhibit the same structure at all levels of detail. Figure 4.42 shows a famous example of such a shape, and its construction. We start with the shape at the top, consisting of four equal line segments, arranged as a straight line with a bump in the middle. We then replace each segment by a scaled down copy of the entire shape, as shown on the second line. We continue in the same way, replacing each of the segments of each of the scaled down copies with a copy of the original shape scaled down further, and so on. The bottom two lines show later stages in the procedure. You can imagine that, if we were able to continue in this way to infinity, we would end up with a crinkly curve, with a bulge in the middle and two lesser crinkly bulges to each side. If

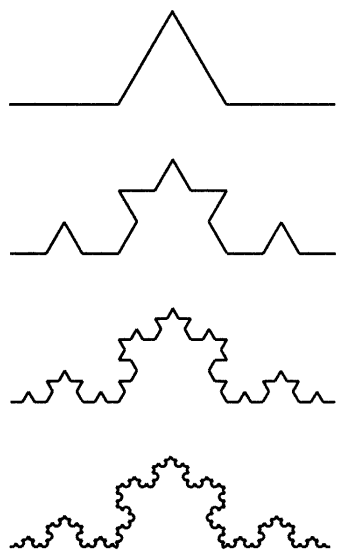


Figure 4.42
Constructing a well-known fractal curve

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 124
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

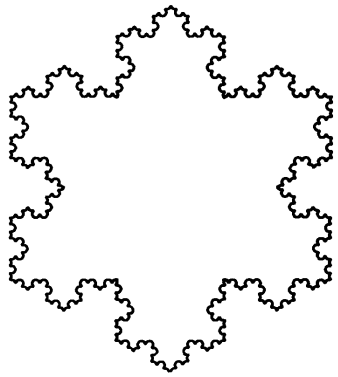


Figure 4.43
A fractal snowflake

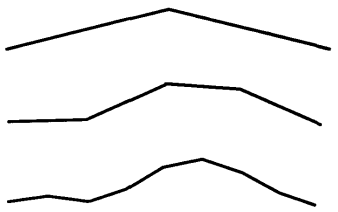


Figure 4.44
Constructing a fractal mountainside

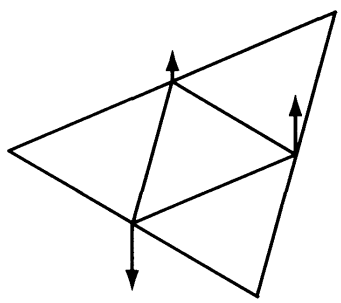


Figure 4.45
3-D random fractal construction

you were then to magnify any part of the curve and look at it you would see a crinkly curve, with a bulge in the middle and two lesser crinkly bulges to each side, which, if you were to magnify any part of it ...

The appearance of certain natural features, such as coastlines, mountains and the edges of clouds, approximates this property: their small scale structure is the same as their large scale structure. Figure 4.43 shows that three of the curves we just constructed can be put together to make a snowflake. Fractals can be extended to three-dimensional structures that exhibit the same sort of similarity at different scales. Whereas this particular sort of structure cannot be described by equations, it is, as we have just demonstrated, easily described by a recursive algorithm.

Where fractal algorithms are used to model natural phenomena, an element of randomness is usually introduced. For example, a very simple fractal might be made by splitting a line into two parts, moving the centre point a certain distance, and then applying the construction recursively to the two halves. If the distance moved is not a constant proportion of the length of the segment but a random distance, the resulting shape will still display a recognizable similarity at different scales, without being exactly internally replicated. Figure 4.44 shows a curve being constructed in the manner just described; it could be said to resemble a mountain slope. In three dimensions, a similar construction can be applied to construct terrains out of internally sub-divided triangular areas. By joining the mid-points of the three sides of a triangle, it can be divided into four smaller triangles. The mid-points can be randomly displaced perpendicularly to the plane of the original triangle, as indicated in Figure 4.45, and then the construction can be applied to each of the small triangles. This process can be applied to arbitrary polygons, and repeated indefinitely to produce arbitrarily fine detail with the irregular appearance of natural terrain, as exhibited by the landscape in Figure 4.46.

➲ Note that the structures used in this sort of modelling are not, strictly mathematically speaking, fractals. A mathematician's fractal shows the same structure at *all* scales, right on to infinity. Fractals are said to be *self-similar*. In computer graphics, we cannot do this, nor do we want to: we can't show any detail smaller than a single pixel, and in any case, to model natural features we must introduce a random element. The randomized structures commonly used in modelling are a finite approximation to a class of structures

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 125
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

known as *random fractals*, which have the property that at any scale the components have the same statistical distribution as the whole structure. This property is a random equivalent of the property of true fractals that at any scale the components are identical to the whole. To ensure such *statistical self-similarity*, constraints must be placed on the random factors introduced at each stage of the construction of these objects.

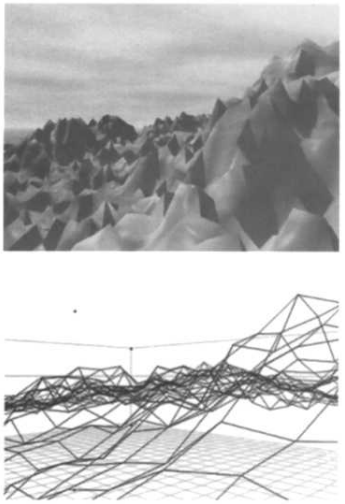


Figure 4.46
Fractal terrain

Two other procedural modelling techniques deserve a brief mention. *Metaballs* are sometimes used to model soft objects. Metaballs are just spheres, but their interaction is modelled using the same sort of mathematics as physicists use to model the electric fields around charged spheres. When two metaballs are placed close together, the fields coalesce into a composite field with a smooth boundary (see Figure 4.47). In 3-D graphics, this boundary is used as the surface of a composite shape. Complex objects can be built by sticking metaballs together. The resulting shapes have a soft, organic quality that is difficult to achieve using other methods of modelling. The process of constructing objects from metaballs somewhat resembles modelling using clay, so tools based on this approach can have a more intuitive feel than more traditional, geometry-based, systems. The fields around other shapes besides spheres can be modelled in a similar way, to allow a wider range of soft objects to be constructed easily. These models count as procedural ones because the disposition of the surfaces is computed algorithmically from the positions of the balls; there is no need to shape the actual surface, in the way polygon meshes must be shaped by the designer.

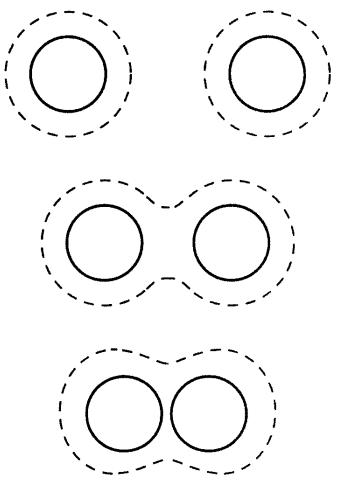


Figure 4.47
Coalescing fields around metaballs

None of the techniques we have described so far is particularly suitable for modelling phenomena such as rain, fountains, fireworks, or grass, consisting of a large number of semi-independent elements, with some shared properties. Often, the elements are moving, and the laws of motion impose certain patterns on them, as in the case of a fountain. To model such things by placing individual drops of water, blades of grass, sparks, and so on, calls for a dedication to detail, and an amount of time, that many 3-D graphic artists lack. *Particle systems* allow features made out of many particles to be specified in terms of a few parameters, from which the positions of the individual particles can be computed algorithmically. Particle systems were first used for film special effects[5] but are now available, at varying levels of sophistication, in desktop 3-D systems.

5 Their first reported use was in creating the 'Genesis effect' in *Star Trek II: The Wrath of Khan.*

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 126
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

The modelling technique used in particle systems leads on to the ultimate procedural modelling system: physics. Established modelling techniques are primarily based on the desired appearance of the objects. If, instead, we base our models on the physical characteristics of objects — their mass and its distribution, elasticity, optical properties, and so on — the appearance can be deduced using physical laws. Such physical models could be tremendously powerful, since a single description of an object would suffice to determine its appearance in any position or environment. Combined with the laws of motion, physical models could be used to describe moving objects, and so to generate animation. Unfortunately, although physicists feel able to construct models of the evolution of the entire universe, modelling such things as the way clothing drapes over a figure is beyond current theories. A great deal of research has been carried out in recent years, particularly into modelling textiles, and it is beginning to bear fruit, but it requires intensive computation and its use is still confined to research laboratories and high-budget film production.

Rendering

3-D models only exist inside computers, all we can see is two-dimensional images generated from them.[6] This is the case both when we wish to produce a final image from a model, and when we are working on a model in a 3-D application. In both cases, a rendering operation is needed; in the latter case, this must be done rapidly enough to provide visual feedback. As we will shortly see, rendering high quality images is time consuming, so it is usually necessary to compromise on quality while a model is being built. Because of the complexity of rendering, and its computational demands, it is common for it to be handled by a specialized module, usually referred to as a *rendering engine*. It may be the case that a rendering engine is optimized for multiprocessing, so that rendering can be speeded up by using a collection of processors operating in parallel.

6
Research work is being done into generating 3-D holographic projections directly from models, but it still has a long way to go.

Almost all contemporary 3-D graphics aspires to the sort of realism associated with photographs. That is, it uses Renaissance perspective conventions (distant objects are smaller than near ones) and attempts to represent the effects of light on different surfaces in a way that is consistent with the laws of optics. The mathematics of perspective projection has been understood for a long time, so it

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 127
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

is relatively easy for a rendering engine to work out which points on an image plane correspond to each of the nodes of a 3-D model. By joining these points to show the edges of objects and of the polygons making up surfaces, it is possible to produce a *wire frame* image of a model, as shown in Figure 4.48. Wire frames are often used as preview images in modelling programs. They can also be useful in computer-aided design systems, since they contain enough information to provide answers to questions such as 'Does the door provide adequate clearance for the piano when it is fully open?' For most purposes, though, they are not acceptable.

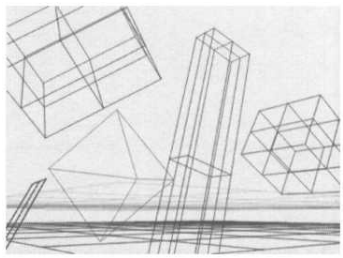


Figure 4.48
Wire frame rendering of objects

Wire frames' most noticeable feature is the absence of any surfaces. This has the obvious disadvantage that no surface detail can be shown, and the less obvious disadvantage that they do not contain enough information to enable us to determine objects' orientations unambiguously — in Figure 4.48, can you determine the form and orientation of every object, and which of their vertices is nearest to you? (Look at Plate 8 to check your answers.) In order to show the surfaces of objects, it is necessary to determine which of them are visible. The far sides of objects are considered hidden in perspective views[7], as are surfaces, or parts of surfaces, with other objects in front of them. Determining which surfaces are visible, starting only from a model based on the coordinates of corners of objects, is not a trivial task, but it is one that has received attention almost from the beginning of computer graphics. There are several tried and tested algorithms for performing *hidden surface removal*. Once this has been done, the next task is to determine how to render the visible surfaces.

7
One could envisage a cubist projection in which this was not the case.

One answer is to colour them arbitrarily, ensuring that no two adjacent faces are the same colour. This approach makes no pretence at photographic realism, but ensures that the shapes of objects are clearly visible. An alternative is to assign a colour to each object, and render its entire surface in that colour. This is more consistent with our expectations about surfaces, but leaves some visual ambiguity. To resolve this ambiguity, and to begin to produce images that resemble photographs, we must take account of the way light interacts with surfaces. This means that our models must include sufficient information about lights and surface properties. In most 3-D modelling programs, surface characteristics can be associated with an object by setting the values of some parameters, including its colour and reflectivity. (Some systems allow the designer to paint on the surface of models in order to specify

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 128
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

their surface characteristics indirectly.) Lights can be created and positioned in the same way as other objects; the designer merely selects a type of light — spotlight or point source (such as the sun), for example — and specifies its position and intensity. It is up to the rendering engine to produce an image that is consistent with the lights and materials that have been used in the model.

Different algorithms — usually called *shading algorithms* — are employed for this task, each incorporating different models of how light interacts with surfaces,[8] and each using different approximations to allow rendering to be performed with reasonable efficiency. Rendering engines usually allow you to choose the most appropriate shading algorithm for the needs of a particular model. There is usually a trade-off between the final quality of the image — or at least, of its treatment of light — and the amount of computation required.

8
The models of illumination are loosely derived from the physics, but also incorporate heuristics aimed at producing an acceptable result and not an optical simulation.

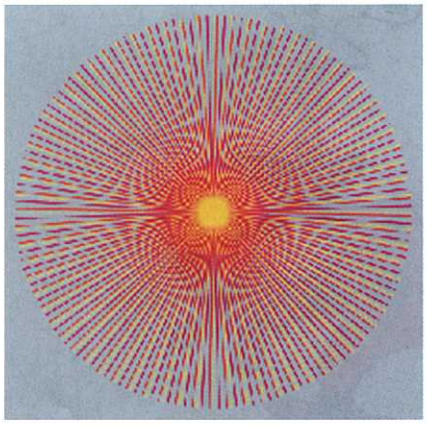
The simplest way of shading an object whose surface is made out of polygons is to calculate a colour value for each polygon, based on the light striking it and its own optical properties, and use that value to colour the whole polygon. Adjacent polygons will often have different colours — they may be at a different angle to the light source, for example — and, as we mentioned earlier, this will make the discontinuities between polygons readily visible. To disguise this effect, and to produce a more convincing rendering of the polygons, more sophisticated algorithms interpolate the colour across each polygon, on the basis of colour values calculated at the vertices. One such interpolation scheme is *Gouraud shading*, named after its inventor. An alternative, which does the interpolation differently, is *Phong shading*. Phong's approach works better when the illumination model being used takes account of *specular reflection* — the light that bounces off the surface of shiny (or partially shiny) objects. The resulting highlights are rendered by Phong shading in a quite convincing way.

The shading algorithms mentioned so far only deal with each object in isolation, but in reality, the appearance of an object may be affected by the presence of others. As an extreme example, the appearance of a glass full of water will depend on what is behind it, while that of a mirror will depend on the objects it reflects. Generally, light may bounce off several objects before reaching the eye, and be influenced by their colour and surface. *Ray tracing* is a shading algorithm that attempts to take account of this. It works by tracing the path of a ray of light back from each pixel in the the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 129
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046



1. An unconventional user interface (Bryce)



2. Moiré patterns



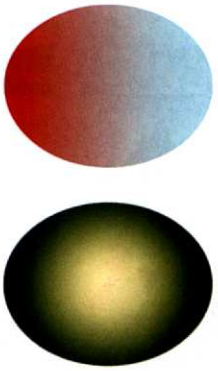
3. Vector image of a poppy



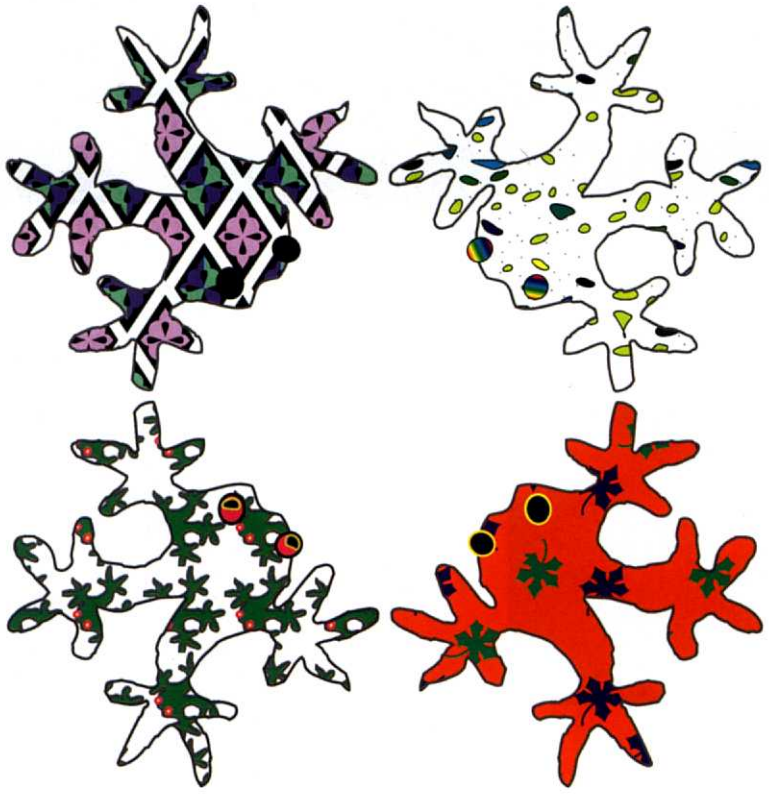
4. Bitmapped image of an iris



6. Use of gradient meshes



5. Gradient fills



7. Pattern fills



8. Ray traced rendering with different lighting and surface characteristics



9. A figure modelled and posed in Poser



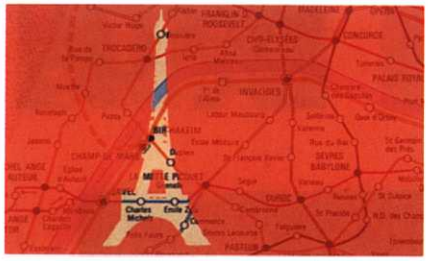
10. A landscape modelled in Bryce







11. Fading with an alpha channel gradient









12. Compositing with an alpha channel





13. A vignette

rendered image to a light source. On the way, the ray may bounce off several surfaces, and the ray tracer will take account of their effect, using a model of illumination and information about the materials of the objects in the scene, to compute the colour and intensity of the starting pixel. Plate 8 shows two renderings of Figure 4.48 by a ray tracer, and illustrates the qualities of different surfaces that can be applied to the objects. The ray tracing computation is quite complex, and it must be repeated for every pixel in the rendered image — very possibly millions of pixels. Until recently, therefore, ray tracing was confined to high-performance workstations, but the extraordinary advance in the speed of personal computers has made it feasible to use ray tracing in desktop 3-D systems. It is now the preferred shading algorithm for 'photo-realistic' graphics. It can produce excellent results, particularly in scenes including transparent or semi-transparent objects.

An alternative approach to the interactions between objects is *radiosity*, which attempts to model the complex reflections that occur between surfaces that are close together. This provides a more accurate representation of scattered and ambient light, and is especially useful for interior scenes. Radiosity is more accurately based on the physics of light than other shading algorithms. It also differs from them in computing the lighting on a model independently of any rendered view of it; essentially, it adds the computed light values to the model. This means that the final rendering of an image can be done very efficiently, although the initial computation of lighting values is slow.

These shading algorithms depend on information about the material of which an object is composed. Where the surface contains a great deal of detail, it may not be feasible to specify this information precisely enough for every small feature to be rendered accurately. A popular way of adding surface details to 3-D models is *texture mapping*. An image, typically a pattern representing some particular sort of surface's appearance, such as fur, bark, sand, marble, and so on, is mathematically wrapped over the surface of the object. That is, mathematical transformations are applied to the pixels of the rendered object, based on those of the image, to produce the appearance of an object with the image wrapped around it. To see the effect of this, imagine cutting a picture out of a magazine, wrapping it round a physical object, and sticking it down. This will work quite well for boxes, producing a box whose surface has the appearance of the picture, but it will work much less

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 134
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

well for spherical or irregular shaped objects. If you can further imagine that the picture was printed on a very thin sheet of rubber, you can see that it would be possible to glue it to these shapes, but that it would become distorted in the process. Texture mapping is similar to gluing rubber pictures to objects, and may introduce the same sort of distortion, but it can also provide a convenient means of applying detail to a surface. Related operations include *bump mapping*, where, instead of using the picture to form the surface appearance, we use it to apply bumpiness or roughness, and *transparency mapping* and *reflection mapping*, which modify the corresponding optical characteristics on the basis of a two-dimensional map in the same way.

⇨ It is by no means uncommon for a rendered image to be imported into an image manipulation program, such as Photoshop, for further processing of the sort we will describe in Chapter 5.

Specialized Types of 3-D Graphics

The complexity of modelling and rendering arbitrary 3-D objects, and the difficulty of mastering general-purpose 3-D tools, strongly suggest that, where possible, specialized tools dedicated to a specific type of model should be used. With a specialized modelling program, a designer can use higher level abstractions that belong to a particular domain without having to provide general modelling tools for any and all types of object. For example, in a figure-modelling program you would build a body out of arms and legs, and so on, instead of trying to construct it as a polygon mesh. The rendering engine can use algorithms that are optimized for the characteristic models produced within the limits set by the modeller. A further potential advantage is that manipulating objects that correspond to things in the world, instead of manipulating their geometrical components, is better fitted to the ways of seeing that artists and designers are experienced with. (Traditional 3-D systems, on the other hand, are generally more suited to the ways of seeing of engineers and scientists, who may find them easier and more familiar than the specialized applications we will describe next.)

Two prominent examples of such tools (both from MetaCreations) are *Bryce*, for landscapes, and *Poser*, for human and animal figures.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 135
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046



Figure 4.49
Bryce's terrain editor

In Bryce, modelling terrain is a basic activity, not, for example, one way of using polygon mesh objects. A terrain object represents the shape of a landscape; it is constructed from a greyscale image, whose brightness represents the height of the 3-D terrain model. Figure 4.49 shows Bryce's terrain editor, with a preview of a terrain, and the image (a fractal image, in this case) from which it is being generated. Terrains can be based on an imported image, or one painted by hand; they can be generated automatically using fractals, or built from US Geological Survey satellite data; they can be eroded, as if by rain, and given features such as ridges or canyons. Elaborate facilities are provided for creating textures and materials to apply to terrains, to simulate the appearance of natural or fantastic phenomena.

To complement its terrain modelling, Bryce provides facilities for modelling sky and atmosphere. Clouds, fog, haze, the sun, moon and stars, even rainbows can be added to the sky. All these features can be customized to a fine degree.

Bryce uses ray tracing, optimized for landscape models, to render its final output. Although Bryce is by no means a trivial program, it allows a designer without the specialized skills required by general 3-D applications to produce landscapes, such as the one shown in Plate 10, with relative ease.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 136
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Poser is dedicated to posing figures in a realistic way. Figures are a digital equivalent of the traditional manikins, or wooden jointed models used by artists as a substitute for live models. The limbs of manikins can be positioned to imitate lifelike poses, which can be drawn by an artist. In a similar way, the limbs and other body parts of a Poser figure can be placed in a pose, and then rendered to produce an image of a figure. Just as the joints of a well-made manikin only move in ways that real people's joints can move, so Poser figures are subject to constraints that force their poses to be physically realizable: the hand on a figure of a person cannot turn 360° on their wrist, for example. Recent releases of Poser provide extensive control over facial expressions as well as poses.

Poser has a library of figures, including variations of male and female human figures, and some animals (although these are not as fully realized as the human figures). Most aspects of their physiology can be customized, and they can be given clothes, which can also be customized; textures and other maps can be applied to people and clothes, either to create realistic skin and textiles, or to produce fantastic effects, such as transparent people. Poser treats figures as figures, not as 3-D models of the kind we described earlier. It thus provides an intuitive way of producing 3-D models representing people and animals. Because of our particularly intimate understanding of figures, and the notorious difficulty of producing convincing representations of them, this is an especially suitable domain for a specialized modelling approach. Whether results such as those shown in Plate 9 have any worth is a matter of opinion. (There exist more powerful figure modelling programs, which are used by film effects units to produce more lifelike results.)

⇨ Poser provides a good example of the need for ingenious computer programs to be supplemented by human skills. In order to produce convincing results, whether in posing figures or in creating new ones, it is necessary to understand how people are put together and how they move. This sort of understanding can only come from lengthy study, as practised in traditional life drawing classes. Although the computer might make it unnecessary to know how to handle charcoal or paint, it does not affect the need for the real skills of seeing and understanding that traditional art education seeks to develop.

Poser and Bryce can be used together. For example, a figure made in Poser can be imported into Bryce to be placed in a landscape. Both applications can also export models in formats

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 137
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

that can subsequently be imported into a general purpose modelling application. Models constructed in both Poser and Bryce can be animated — see Chapter 11.

⇨ Another of MetaCreations' programs is specialized in a different way. *Canoma* is dedicated to a single way of constructing models: *image-assisted* modelling. The starting point for a model is a photograph (or several photographs), and the model is constructed by 'pinning' the nodes of a wire frame to corresponding points on the photograph. The photographic image is then mapped on to the surface of the resulting solid, so, in effect, the original picture is extended into the third dimension. Clearly, in order to produce an accurate model, pictures of at least two sides of the object must be used in conjunction with each other. The pinning operation works best on objects, such as buildings and domestic appliances, which are fundamentally simple geometric solids. Hence, although Canoma is not dedicated to a specific area, in the way that Bryce and Poser are, this method of modelling is best suited for certain applications, such as producing architectural models.

Further Information

Animation of vector graphics, including 3-D, is described in Chapter 11. [FvDFH96] contains a comprehensive technical account of most aspects of vector graphics; [WW92] provides details of ray tracing and radiosity. [Vin92] and [O'R98] are more accessible introductions to 3-D graphics.

Exercises

1. True or false: anti-aliasing never affects the appearance of vertical and horizontal lines?
2. (For mathematicians.) For many purposes, it is convenient to know the *bounding box* of a vector graphics object, that is, the smallest rectangle that entirely encloses the object.
   (a) Show that the bounding box on its own is an adequate representation for an ellipse.
   (b) Can a Bézier curve be represented by a bounding box?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 138
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

3. What happens if two Bézier curves with control points P_1, P_2, P_3 and P_4, and P_4, P_5, P_6 and P_7 are joined at P_4 so that P_3, P_4 and P_5 are in a straight line but P_5 and P_3 are on the same side of P_4?

4. Gradient fills are not directly provided in PostScript and other low-level graphics languages. Describe how you would implement linear and radial gradients using the other vector graphics primitives described in this chapter.

5. (a) The algorithm for computing the winding number of a point relative to a path does not tell you what to do if the line you construct coincides with part of the path or touches it tangentially. How would you cope with these possibilities?

   (b) An alternative to the non-zero winding number rule that is sometimes used instead is the *odd-even rule*, which is computed in a similar way by constructing a line from a point and counting crossings, only this time you just count the number of times the path crosses the line. If it is even, the point is outside the path, if it is odd, it is inside. Does this rule always give the same result as the non-zero winding number rule? If not, when will they disagree?

6. The difference operation in constructive solid geometry is not commutative, that is, if we subtract shape A from shape B the result is different from what we get by subtracting shape B from shape A. Figure 4.40 shows the effect of subtracting the cylinder from the ellipsoid. Sketch the object that would be obtained by subtracting the ellipsoid from the cylinder.

7. Under what circumstances would you *not* use ray tracing to render a 3-D scene?

8. Suggest two types of specialized 3-D graphics, besides landscapes and figures, for which it would be sensible to use special-purpose tools instead of general 3-D graphics packages. Outline the main features you would expect in the corresponding tools — what abstract components would they support, and how might they take advantage of characteristics of your problem domains to optimize rendering performance?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 139
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Bitmapped Images

5

Conceptually, bitmapped images are much simpler than vector graphics. There is no need for any mathematical modelling of shapes, we merely record the value of every pixel in the image. This means that when the image is created, a value has to be assigned to every pixel, but many images are created from external sources, such as scanners or digital cameras, which operate in a bitmapped fashion anyway. For creating original digital images, programs such as Painter allow visual artists to use familiar techniques (at least metaphorically) to paint images. As we pointed out earlier, the main cost for this simplicity is in the size of image files. There is, though, one area where bitmaps are less simple than vectors, and that is resolution. We touched on this in Chapter 3, but now we shall look at it in more detail.

Resolution

The concept of resolution is a simple one, but the different ways in which the word is used can be confusing. Resolution is a measure of how finely a device approximates continuous images using finite

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 140
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

pixels. It is thus closely related to sampling, and some of the ideas about sampling rates introduced in Chapter 2 are relevant to the way resolution affects the appearance of images.

There are two common ways of specifying resolution. For printers and scanners, the resolution is usually stated as the number of dots per unit length. Usually, in English-speaking countries, it is specified anachronistically in units of dots per inch (dpi). At the time of writing, desktop printers typically have a resolution of 600 dpi, while imagesetters (as used for book production) have a resolution of about 1200 to 2700 dpi; flatbed scanners' resolution ranges from 300 dpi at the most basic level to 3600 dpi; transparency scanners and drum scanners used for high quality work have even higher resolutions.

In video, resolution is normally specified by giving the size of a frame, measured in pixels — its *pixel dimensions*. For example, a PAL frame is 768 by 576 pixels; an NTSC frame is 640 by 480.[1] Obviously, if you know the physical dimensions of your TV set or video monitor, you can translate resolutions specified in this form into dots per inch. For video it makes more sense to specify image resolution in the form of the pixel dimensions, because the same pixel grid is used to display the picture on any monitor (using the same video standard) irrespective of its size. Similar considerations apply to digital cameras, whose resolution is also specified in terms of the pixel dimensions of the image.

1 Sort of — see Chapter 10.

Knowing the pixel dimensions, you know how much detail is contained in the image; the number of dots per inch on the output device tells you how big that same image will be, and how easy it will be to see the individual pixels.

Computer monitors are based on the same technology as video monitors, so it is common to see their resolution specified as an image size, such as 640 by 480 (for example, VGA), or 1024 by 768. However, monitor resolution is sometimes quoted in dots per inch, because of the tendency in computer systems to keep this value fixed and to increase the pixel dimensions of the displayed image when a larger display is used. Thus, a 14 inch monitor provides a 640 by 480 display at roughly 72 dpi; a 17 inch monitor will provide 832 by 624 pixels at the same number of dots per inch.

⇨ There is an extra complication with colour printers. As we will see in Chapter 6, in order to produce a full range of colours using just four or six inks, colour printers arrange dots in groups, using a pattern of

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 141
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

different coloured inks within each group to produce the required colour by optical mixing. Hence, the size of the coloured pixel is greater than the size of an individual dot of ink. The resolution of a printer taking account of this way of mixing colours is quoted in *lines per inch* (or other unit of length), following established printing practice.[2] The number of lines per inch will be as much as five times lower than the number of dots per inch — the exact ratio depends on how the dots are arranged, which will vary between printers, and may be adjustable by the operator. You should realise that, although a colour printer may have a resolution of 1200 *dots* per inch, this does not mean that you need to use such a high resolution for your images. A line resolution of 137 per inch is commonly used for printing magazines; the colour plates in this book are printed at a resolution of 150 lines per inch.

2 This figure is sometimes called the *screen ruling*, again following established terminology from the traditional printing industry.

Now consider bitmapped images. An image is an array of pixel values, so it necessarily has pixel dimensions. Unlike an input or output device, it has no *physical* dimensions. In the absence of any further information, the physical size of an image when it is displayed will depend on the resolution of the device it is to be displayed on. For example, the square in Figure 3.1 on page 70 is 128 pixels wide. When displayed at 72 dpi, as it will be on a Macintosh monitor, for example, it will be 45mm square. Displayed without scaling on a higher resolution monitor at 115 dpi, it will only be a little over 28mm square. Printed on a 600 dpi printer, it will be about 5mm square. (Many readers will probably have had the experience of seeing an image appear on screen at a sensible size, only to be printed the size of a postage stamp.) In general, we have:

physical dimension = *pixel dimension*/*device resolution*

where the device resolution is measured in pixels per unit length. (If the device resolution is specified in pixels per inch, the physical dimension will be in inches, and so on.)

Images have a natural size, though: the size of an original before it is scanned, or the size of canvas used to create an image in a painting program. We often wish the image to be displayed at its natural size, and not shrink or expand with the resolution of the output device. In order to allow this, most image formats record a resolution with the image data. This resolution is usually quoted in units of *pixels* per inch (ppi), to distinguish it from the resolution of physical devices. The stored resolution will usually be that of the device from which the image originated. For example, if the image is scanned at 600 dpi, the stored *image resolution*

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 142
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

will be 600 ppi. Since the pixels in the image were generated at this resolution, the physical dimensions of the image can be calculated from the pixel dimensions and the image resolution. It is then a simple matter for software that displays the image to ensure that it appears at its natural size, by scaling it by a factor of *device resolution/image resolution*. For example, if a photograph measured 6 inches by 4 inches, and it was scanned at 600dpi, its bitmap would be 3600 by 2400 pixels in size. Displayed in a simple-minded fashion on a 72dpi monitor, the image would appear 50 inches by 33.3 inches (and, presumably, require scroll bars). To make it appear at the desired size, it must be scaled by 72/600 = 0.12, which, as you can easily verify, reduces it to its original size.

If an image's resolution is lower than that of the device on which it is to be displayed, it must be scaled up, a process which will require the interpolation of pixels. This can never be done without loss of image quality, so you should try to ensure that any images you use in your multimedia productions have an image resolution at least as high as the monitors on which you expect your work to be viewed.

If, on the other hand, the image's resolution is higher than that of the output device, pixels must be discarded when the image is scaled down for display at its natural size. This process is called *downsampling*. Here we come to an apparent paradox. The subjective quality of a high resolution image that has been downsampled for display at a low resolution will often be better than that of an image whose resolution is equal to the display resolution. For example, when a 600 dpi scan is displayed on screen at 72 dpi, it will often look better than a 72 dpi scan, even though there are no more pixels in the displayed image. This is because the scanner samples discrete points; at lower resolutions these points are more widely spaced than at high resolutions, so some image detail is missed. The high resolution scan contains information that is absent in the low resolution one, and this information can be used when the image is downsampled for display. For example, the colour of a pixel might be determined as the average of the values in a corresponding block, instead of the single point value that is available in the low resolution scan. This can result in smoother colour gradients and less jagged lines for some images. The technique of sampling images (or any other signal) at a higher resolution than that at which it is ultimately displayed is called *oversampling*.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 143
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

The apparently superior quality of oversampled images will only be obtained if the software performing the downsampling does so in a sufficiently sophisticated way to make use of the extra information available in the high-resolution image. Web browsers are notoriously poor at down-sampling, and usually produce a result no better than that which would be obtained by starting from a low-resolution original. For that reason, images intended for the World Wide Web should be downsampled in advance using a program such as Photoshop.

We will describe resampling in more detail later in this chapter, when we consider applying geometrical transformations to bit-mapped images.

Information once discarded can never be regained. The conclusion would seem to be that one should always keep high resolution bit-mapped images, downsampling only when it is necessary for display purposes or to prepare a version of the image for display software that does not downsample well. However, the disadvantage of high resolution images soon becomes clear. High resolution images contain more pixels and thus occupy more disk space and take longer to transfer over networks. The size of an image increases as the square of the resolution, so, despite the possible gains in quality that might come from using high resolutions, in practice we more often need to use the *lowest* resolution we can get away with. This must be at least as good as an average monitor if the displayed quality is to be acceptable. Even at resolutions as low as 72 or 96 ppi, image files can become unmanageable, especially over networks. To reduce their size without unacceptable loss of quality we must use techniques of data compression.

Image Compression

Consider again Figure 3.1. We stated on page 70 that its bitmap representation required 16 kilobytes. This estimate was based on the assumption that the image was stored as an array, with one byte per pixel. In order to display the image or manipulate it, it would have to be represented in this form, but when we only wish to record the values of its pixels, for storage or transmission over a network, we can use a much more compact data representation. Instead of storing the value of each pixel explicitly, we could instead store

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 144
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

a value, followed by a count to indicate a number of consecutive pixels of that value. For example, the first row consists of 128 pixels, all of the same colour, so instead of using 128 bytes to store that row, we could use just two, one to store the value corresponding to that colour, another to record the number of occurrences. Indeed, if there was no advantage to be gained from preserving the identity of individual rows, we could go further, since the first 4128 pixels of this particular image are all the same colour. These are followed by a run of 64 pixels of another colour, which again can be stored using a count and a colour value in two bytes, instead of as 64 separate bytes all of the same value.

This simple technique of replacing a run of consecutive pixels of the same colour by a single copy of the colour value and a count of the number of pixels in the run is called *run-length encoding (RLE)*. In common with other methods of compression, it requires some computation in order to achieve a saving in space. Another feature it shares with other methods of compression is that its effectiveness depends on the image that is being compressed. In this example, a large saving in storage can be achieved, because the image is extremely simple and consists of large areas of the same colour, which give rise to long runs of identical pixel values. If, instead, the image had consisted of alternating pixels of the two colours, applying RLE in a naïve fashion would have led to an increase in the storage requirement, since each 'run' would have had a length of one, which would have to be recorded in addition to the pixel value. More realistically, images with continuously blended tones will not give rise to runs that can be efficiently encoded, whereas images with areas of flat colour will.

It is a general property of any compression scheme that there will be some data for which the 'compressed' version is actually larger than the uncompressed. This must be so: if we had an algorithm that could always achieve some compression, no matter what input data it was given, it would be possible to apply it to its own output to achieve extra compression, and then to the new output, and so on, arbitrarily many times, so that any data could be compressed down to a single byte (assuming we do not deal with smaller units of data). Even though this is clearly absurd, from time to time people claim to have developed such an algorithm, and have even been granted patents.

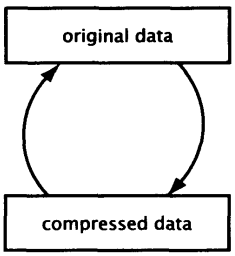


Figure 5.1
Lossless compression

Run-length encoding has an important property: it is always possible to decompress run-length encoded data and retrieve an

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 145
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

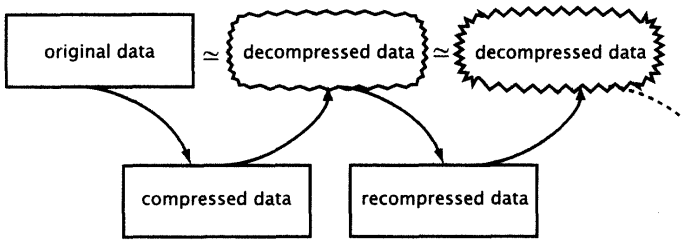


Figure 5.2
Lossy compression

exact copy of the original data as it was before compression. If we take the 16 kilobyte array representing Figure 3.1 and apply RLE compression to it, we will be able to apply an obvious decompression algorithm to the result to get back the original array. RLE is an example of a *lossless* compression technique, since no information is lost during a compression/decompression cycle (see Figure 5.1). In contrast, *lossy* compression techniques work by discarding some information during the compression process. Once discarded, information can never be retrieved, so when data that has been lossily compressed is decompressed, the result is only an approximation to the original data (see Figure 5.2). Lossy compression is well suited to data, such as images and sound, that originates in analogue form, since, as we saw in Chapter 2, its digital representation is an approximation anyway. Clever design of compression algorithms can ensure that only data that is insignificant to perception of the image or sound is discarded, so that substantial savings in storage can be achieved with little or no perceived loss in quality. However, because information is lost during every compression/decompression cycle, if data is repeatedly compressed, decompressed and then compressed again, its quality will progressively deteriorate.

A full description of the mechanics of compression lies beyond the scope of this book; we will just introduce the main algorithms you are likely to encounter.

Lossless Compression

RLE is the simplest lossless compression algorithm to understand, but it is far from the most effective. The more sophisticated lossless algorithms you are likely to encounter fall into two classes. Algorithms of the first class work by re-encoding data so that the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 146
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

most frequent values occupy the fewest bits. For example, if an image uses 256 colours, each pixel would normally occupy eight bits (see Chapter 6). If, however, we could assign codes of different lengths to the colours, so that the code for the most common colour was only a single bit long, two-bit codes were used for the next most frequent colours, and so on, a saving in space would be achieved for most images. This approach to encoding, using *variable-length codes*, dates back to the earliest work on data compression and information theory, carried out in the late 1940s. The best known algorithm belonging to this class is *Huffman coding*.[3]

3
You will find Huffman coding described in many books on data structures, because its implementation provides an interesting example of the use of trees.

Although Huffman coding and its derivatives are still used as part of other, more complex, compression techniques, since the late 1970s variable-length coding schemes have been superseded to a large extent by *dictionary-based* compression schemes. Dictionary-based compression works by constructing a table, or dictionary, into which are entered strings of bytes (not necessarily corresponding to characters) that are encountered in the input data; all occurrences of a string are then replaced by a pointer into the dictionary. The process is similar to the tokenization of names carried out by the lexical analyser of a compiler using a symbol table. In contrast to variable-length coding schemes, dictionary-based schemes use fixed-length codes, but these point to variable-length strings in the dictionary. The effectiveness of this type of compression depends on choosing the strings to enter in the dictionary so that a saving of space is produced by replacing them by their codes. Ideally, the dictionary entries should be long strings that occur frequently.

Two techniques for constructing dictionaries and using them for compression were described in papers published in 1977 and 1978 by two researchers called Abraham Lempel and Jacob Ziv, thus the techniques are usually called LZ77 and LZ78.[4] A variation of LZ78, devised by another researcher, Terry Welch, and therefore known as LZW compression, is one of the most widely used compression methods, being the basis of the Unix `compress` utility and of GIF files. The difference between LZ77 and LZ78 lies in the way in which the dictionary is constructed, while LZW is really just an improved implementation for LZ77. LZW compression has one drawback: as we mentioned in Chapter 3, it is patented by Unisys, who charge a licence fee for its use. As a result of this, the compression method used in PNG files is based on the legally unencumbered LZ77, as are several widely used general-purpose compression programs, such as PKZIP.

4
It is a curious, but probably insignificant, fact that the papers' authors were actually given as Ziv and Lempel, not the other way round, but the algorithms are never referred to as ZL7x.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 147
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

JPEG Compression

Lossless compresssion can be applied to any sort of data; it is the *only* sort of compression that can be applied to certain sorts, such as binary executable programs, spreadsheet data, or text, since corruption of even one bit of such data may invalidate it. Image data, though, can tolerate a certain amount of data loss, so lossy compression can be used effectively for images. The most important lossy image compression technique is *JPEG compression*. JPEG stands for the Joint Photographic Experts Group,[5] which draws attention to a significant feature of JPEG compression: it is best suited to photographs and similar images which are characterised by fine detail and continuous tones — the same characteristics as bitmapped images exhibit in general.

5
'Joint' refers to the fact that JPEG is a collaboration between two standards bodies, ISO and CCITT (now ITU).

In Chapter 2, we considered the brightness or colour values of an image as a signal, which could be decomposed into its constituent frequencies. One way of envisaging this is to forget that the pixel values stored in an image denote colours, but merely consider them as the values of some variable z; each pixel gives a z value at its x and y coordinates, so the image defines a three-dimensional shape. (See Figure 5.3, which shows a detail from the iris image, rendered as a 3-D surface, using its brightness values to control the height.) Such a shape can be considered as a complex 3-D waveform. We also explained that any waveform can be transformed into the frequency domain using the Fourier Transform operation. Finally, we pointed out that the high frequency components are associated with abrupt changes in intensity. An additional fact, based on extensive experimental evidence, is that people do not perceive the effect of high frequencies very accurately, especially not in colour images.

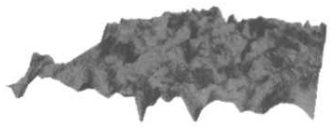


Figure 5.3
Pixel values interpreted as height

Up until now, we have considered the frequency domain representation only as a way of thinking about the properties of a signal. JPEG compression works by actually transforming an image into its frequency components. This is done, not by computing the Fourier Transform, but using a related operation called the *Discrete Cosine Transform (DCT)*. Although the DCT is defined differently from the Fourier Transform, and has some different properties, it too analyses a signal into its frequency components. In computational terms, it takes an array of pixels and produces an array of coefficients, representing the amplitude of the frequency

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 148
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

components in the image. Since we start with a two-dimensional image, whose intensity can vary in both the x and y directions, we end up with a two-dimensional array of coefficients, corresponding to spatial frequencies in these two directions. This array will be the same size as the image's array of pixels.

Applying a DCT operation to an image of any size is computationally expensive, so it is only with the widespread availability of powerful processors that it has been practical to perform this sort of compression — and, more importantly, decompression — without the aid of dedicated hardware. Even now, it remains impractical to apply DCT to an entire image at once. Instead, images are divided into 8×8 pixel squares, each of which is transformed separately.

➲ The DCT of an $N \times N$ pixel image $[p_{uv}, 0 \le u < N, 0 \le v < N]$ is an array of coefficients $[\mathrm{DCT}_{uv}, 0 \le u < N, 0 \le v < N]$ given by

$$\mathrm{DCT}_{uv} = \frac{1}{\sqrt{2N}} C_u C_v \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p_{xy} \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

where

$$C_u, C_v = \begin{cases} 1/\sqrt{2} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

On its own, this doesn't tell you much about why the DCT coefficients are in fact the coefficients of the frequency components of the image, but it does indicate something about the complexity of computing the DCT. For a start, it shows that the array of DCT coefficients is the same size as the original array of pixels. It also shows that the computation must take the form of a nested double loop, iterating over the two dimensions of the array, so the computational time for each coefficient is proportional to the image size in pixels, and the entire DCT computation is proportional to the square of the size. While this is not intractable, it does grow fairly rapidly as the image's size increases, and bitmapped images measuring thousands of pixels in each direction are not uncommon. Applying the DCT to small blocks instead of the entire image reduces the computational time by a significant factor.

Since the two cosine terms are independent of the value of p_{ij}, they can be precomputed and stored in an array to avoid repeated computations. This provides an additional optimization when the DCT computation is applied repeatedly to separate blocks of an image.

Transforming an image into the frequency domain does not, in itself, perform any compression. It does, however, change the data

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 149
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

into a form which can be compressed in a way that minimizes the perceptible effect of discarding information, because the frequency components are now explicitly separated. This allows information about the high frequencies, which do not contribute much to the perceived quality of the image, to be discarded. This is done by distinguishing fewer different possible values for higher frequency components. If, for example, the value produced by the DCT for each frequency could range from 0 to 255, the lowest frequency coefficients might be allowed to have any integer value within this range; slightly higher frequencies might only be allowed to take on values divisible by 4, while the highest frequencies might only be allowed to have the value 0 or 128. Putting this another way, the different frequencies are quantized to different numbers of levels, with fewer levels being used for high frequencies. In JPEG compression, the number of quantization levels to be used for each frequency coefficient can be specified separately in a quantization matrix, containing a value for each coefficient.

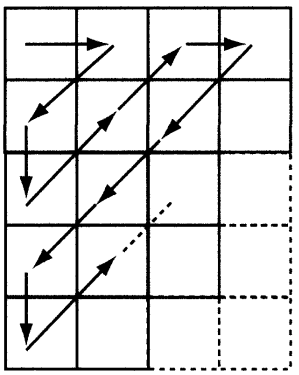


Figure 5.4
The zig-zag sequence

This quantization process reduces the space needed to store the image in two ways. First, after quantization, many components will end up with zero coefficients. Second, fewer bits are needed to store the non-zero coeficients. To take advantage of the redundancy which has thus been generated in the data representation, two lossless compression methods are applied to the array of quantized coefficients. Zeroes are run-length encoded; Huffman coding is applied to the remaining values. In order to maximize the length of the runs of zeroes, the coefficients are processed in what is called the *zig-zag sequence*, as shown in Figure 5.4. This is effective because the frequencies increase as we move away from the top left corner of the array in both directions. In other words, the perceptible information in the image is concentrated in the top left part of the array, and the likelihood is that the bottom right part will be full of zeroes. The zig-zag sequence is thus likely to encounter long runs of zeroes, which would be broken up if the array were traversed more conventionally by rows or columns.

Decompressing JPEG data is done by reversing the compression process. The runs are expanded and the Huffman encoded coefficients are decompressed and then an *Inverse Discrete Cosine Transform* is applied to take the data back from the frequency domain into the spatial domain, where the values can once again be treated as pixels in an image. The inverse DCT is defined very similarly to the DCT itself; the computation of the inverse transform

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 150
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

requires the same amount of time as that of the forward transform, so JPEG compression and decompression take roughly the same time.[6] Note that there is no 'inverse quantization' step. The information that was lost during quantization is gone forever, which is why the decompressed image only approximates the original. Generally, though, the approximation is a good one.

6 On the same machine.

One highly useful feature of JPEG compression is that it is possible to control the degree of compression, and thus the quality of the compressed image, by altering the values in the quantization matrix. Programs that implement JPEG compression allow you to choose a quality setting, so that you can choose a suitable compromise between image quality and compression. You should be aware that, even at the highest quality settings, JPEG compression is still lossy in the literal sense that the decompressed image will not be an exact bit-for-bit duplicate of the original. It will often, however, be visually indistinguishable, but that is not what we mean by 'lossless'.

⇨ The JPEG standard does define a lossless mode, but this uses a completely different algorithm from the DCT-based method used by lossy JPEG. Lossless JPEG compression has never been popular. It has been superseded by a new standard JPEG-LS, but this, too, shows little sign of being widely adopted.

Yet another JPEG standard is also under construction: JPEG2000 aims to be the image compression standard for the new century (or at least the first ten years of it). The ISO's call for contributions to the JPEG2000 effort describes its aims as follows:

> "[...] to create a new image coding system for different types of still images (bi-level, gray-level, color) with different characteristics (natural images, scientific, medical, remote sensing imagery, text, rendered graphics, etc.) allowing different imaging models (client/server, real-time transmission, image library archival, limited buffer and bandwidth resources, etc.) preferably within a unified system."

JPEG compression is highly effective when applied to the sort of images for which it is designed: photographic and scanned images with continuous tones. Such images can be compressed to as little as 5% of their original size without apparent loss of quality. Lossless compression techniques are nothing like as effective on such images. Still higher levels of compression can be obtained by using a lower quality setting, i.e. by using coarser quantization that discards more information. When this is done,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 151
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

the boundaries of the 8×8 squares to which the DCT is applied tend to become visible, because the discontinuities between them mean that different frequency components are discarded in each square. At low compression levels (i.e. high quality settings) this does not matter, since enough information is retained for the common features of adjacent squares to produce appropriately similar results, but as more and more information is discarded, the common features becomes lost and the boundaries show up.

Such unwanted features in a compressed image are called *compression artefacts*. Other artefacts arise when an image containing sharp edges is compressed by JPEG. Here, the smoothing that is the essence of JPEG compression is to blame; sharp edges come out blurred. This is rarely a problem with the photographically originated material for which JPEG is intended, but can be a problem if images created on a computer are compressed. In particular, if text, especially small text, occurs as part of an image, JPEG is likely to blur the edges, often making the text unreadable. For images with many sharp edges, JPEG should be avoided. Instead, images should be saved in a format such as PNG, which uses lossless LZ77 compression.

Image Manipulation

A bitmapped image explicitly stores a value for every pixel, so we can, if we wish, alter the value of any pixel or group of pixels to change the image. The sheer number of pixels in most images means that editing them individually is both time-consuming and confusing — how is one to judge the effect on the appearance of the whole image of particular changes to certain pixels? Or which pixels must be altered, and how, in order to sharpen the fuzzy edges in an out-of-focus photograph? In order for image editing to be convenient, it is necessary that operations be provided at a higher level than that of altering a single pixel. Many useful operations can be described by analogy with pre-existing techniques for altering photographic images, in particular, the use of filters and masks.

Before we describe how images can be manipulated, we ought first to examine why one might wish to manipulate them. There are two broad reasons: one is to correct deficiencies in an image, caused by poor equipment or technique used in its creation or digitization,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 152
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

the other is to create images that are difficult or impossible to make naturally. An example of the former type is the removal of 'red-eye', the red glow apparently emanating from the eyes of a person whose portrait has been taken face-on with a camera using a flash set too close to the lens.[7] Consumer-oriented image manipulation programs often provide commands that encapsulate a sequence of manipulations to perform common tasks, such as red-eye removal, with a single key stroke. The manipulations performed for the second reason generally fall into the category of special effects, such as creating a glow around an object. Image manipulation programs, such as Photoshop, generally supply a set of built-in filters and effects, and a kit of tools for making selections and low-level adjustments. Photoshop uses an open architecture that allows third parties to produce plug-ins which provide additional effects and tools — the development of Photoshop plug-ins has become an industry in its own right, and many other programs can now use them.

7
Red-eye is caused by light reflecting off the subject's retinas.

Many of the manipulations commonly performed on bitmapped images are concerned purely with preparing digital images for print, and are therefore not relevant to multimedia. An operation that is typical of multimedia work, on the other hand, is the changing of an image's resolution or size (which, as we saw previously, are essentially equivalent operations). Very often, images which are required for display on a monitor originate at higher resolutions and must be down-sampled; an image's size may need adjusting to fit a layout, such as a Web page.

⇨ Photoshop is, without doubt, the leading application for image manipulation, being a *de facto* industry standard, and we will use it as our example. There are other programs, though, in particular a powerful package known as *The Gimp*, which has similar capabilities and a similar set of tools to Photoshop's, is distributed under an open software licence for Unix systems. Both of these programs include many features concerned with preparing images for print, which are not relevant to multimedia. Recently, a number of packages, such as Adobe's ImageReady and Macromedia's Fireworks, which are dedicated to preparation of images for the World Wide Web have appeared; these omit print-oriented features, replacing them with others more appropriate to work on the Web, such as facilities for slicing an image into smaller pieces to accelerate downloading, or adding hotspots to create an 'image map' (see Chapter 13).

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 153
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Selections, Masks and Alpha Channels

As we have repeatedly stressed, a bitmapped image is not stored as a collection of separate objects, it is just an array of pixels. Even if we can look at the picture and see a square or a circle, say, we cannot select that shape when we are editing the image with a program, the way we could if it were a vector image, because the shape's identity is not part of the information that is explicitly available to the program, it is something our eyes and brain have identified. Some other means must be employed to select parts of an image when it is being manipulated by a mere computer program.

Ironically, perhaps, the tools that are used to make selections from bitmapped images are more or less the same tools that are used to draw shapes in vector graphics. Most selections are made by drawing around an area, much as a traditional paste-up artist would cut out a shape from a printed image using a scalpel. The simplest selection tools are the rectangular and elliptical *marquee* tools, which let you select an area by dragging out a rectangle or ellipse, just as you would draw these shapes in a drawing program. It is important to realize that you are not drawing, though, you are defining an area within the image.

More often than not, the area you wish to select will not be a neat rectangle or ellipse. To accommodate irregular shapes, thinly disguised versions of the other standard drawing tools may be used: the lasso tool is a less powerful version of Illustrator's pencil tool, which can be used to draw freehand curves around an area to be selected; the polygon lasso is like a pen tool used to draw polylines, rather than curves; a fully-fledged Bézier drawing pen is also available. These tools allow selections to be outlined with considerable precision and flexibility, although their use can be laborious. To ease the task of making selections, two tools are available that make use of pixel values to help define the selected area. These are the magic wand and the magnetic lasso.



Figure 5.5
Selecting with the magic wand

The magic wand is used to select areas on the basis of their colour. With this tool selected, clicking on the image causes all pixels adjacent to the cursor which are similar in colour to the pixel under the cursor to be selected. Figure 5.5 shows an example of the magic wand selecting a highly irregular shape. The tolerance, that is, the amount by which a colour may differ but still be considered sufficiently similar to include in the selection, may be specified. The

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 154
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

magnetic lasso works on a different principle. Like the other lasso tools, it is dragged around the area to be selected, but instead of simply following the outline drawn by the user, it adjusts itself so that the outline snaps to edges within a specified distance of the cursor. Any sufficiently large change in contrast is considered to be an edge. Both the distance within which edges are detected, and the degree of contrast variation that is considered to constitute an edge may be specified. Where an image has well-defined edges, for example, both of these can be set to a high value, so that drawing roughly round an object will cause it to be selected as the outline snaps to the high contrast edges. Where the edges are less well defined, it will be necessary to allow a lower contrast level to indicate an edge, and consequently the outline will have to be drawn with more care, using a narrower detection width.

These semi-automatic selection tools can be somewhat erratic, and cannot generally cope with such demanding tasks as selecting hair or delicate foliage. It is often necessary to make a preliminary selection with one of these tools, and then refine it. The outline of a selection is essentially a vector path, and adjustments are made by moving, adding or deleting control points.

Once a selection has been made, using any of the tools just described, any changes you make to the image, such as applying filters, are restricted to the pixels within the selected area. Another way of describing this is to say that the selection defines a *mask* — the area that is not selected, which is protected from any changes. Image manipulation programs allow you to store one or more masks with an image, so that a selection can be remembered and used for more than one operation — an ordinary selection is ephemeral, and is lost as soon as a different one is made.

The technique of masking off parts of an image has long been used by artists and photographers, who use physical masks and stencils to keep out light or paint. A cardboard stencil, for example, either completely allows paint through or completely stops it. We could store a digital mask with similar 'all or nothing' behaviour by using a single bit for each pixel in the image, setting it to one for all the masked out pixels, and to zero for those in the selection. Thus, the mask is itself an array of pixels, and we can think of it as being another image. If just one bit is used for each pixel, this image will be purely monochromatic. By analogy with photographic masks, the white parts of the image are considered transparent, the black ones opaque.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 155
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Digital masks have properties which are difficult to realize with physical media. By using more than one bit, so that the mask becomes a greyscale image, we can specify different degrees of transparency. For reasons which will be elaborated on in Chapter 6, a greyscale mask of this sort is often called an *alpha channel*. Any painting, filtering or other modifications made to pixels covered by semi-transparent areas of the mask will be applied in a degree proportional to the value stored in the alpha channel. It is common to use eight bits for each pixel of a mask, allowing for 256 different transparency values.

To return to the analogy of a stencil, an alpha channel is like a stencil made out of a material that can allow varying amounts of paint to pass through it, depending on the transparency value at each point. One use for such a stencil would be to produce a soft edge around a cut out shape. In a similar way, the edge of a selection can be 'feathered', which means that the hard transition from black to white in the alpha channel is replaced by a gradient, passing through intermediate grey values, which correspond to partial masking. Any effects that are applied will fade over this transitional zone, instead of stopping abruptly at the boundary. A less drastic way of exploiting alpha channels is to apply anti-aliasing to the edge of a mask, reducing the jagged effect that may otherwise occur. Although anti-aliasing resembles feathering over a very narrow region, the intention is quite different: feathering is supposed to be visible, causing effects to fade out, whereas anti-aliasing is intended to unobtrusively conceal the jagged edges of the selection.

Normally, if an image is pasted into another, it obscures everything underneath it, by overwriting the pixels. However, if the pasted image has an alpha channel, its transparency values are used to mix together the two images, so that the original will show through in the transparent areas of the mask. The value of a pixel p in the resulting composited image is computed as $p = \alpha p_1 + (1 - \alpha)p_2$, where p_1 and p_2 are the values of pixels in the two original images, and α is normalized to lie between 0 and 1.[8] Some familiar effects can be achieved by constructing a mask by hand. For example, if we use a gradient as the mask, we can make one image fade into another. Plate 11 shows an example: the mask shown in Figure 5.6 is used to combine a colour and a black and white version of the same photograph, producing a coloured 'magic corridor' down the middle of the composite image. Another example is shown in Plate 12.

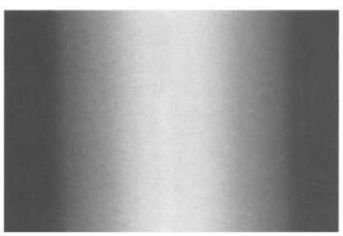


Figure 5.6
Alpha channel for Plate 11

8 That is, if the α value is stored in 8-bits we divide it by 255.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 156
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Here, a mask has been created by saving a magic wand selection from the photograph of the Eiffel tower. This mask has then been attached to the map of Paris and used to combine it with the French flag.

Pixel Point Processing

Image processing is performed by computing a new value for each pixel in an image. The simplest methods compute a pixel's new value solely on the basis of its old value, without regard to any other pixel. So for a pixel with value p, we compute a new value $p' = f(p)$, where f is some function, which we will call the *mapping function.* Such functions perform *pixel point processing.* A simple, if only rarely useful, example of pixel point processing is the construction of a negative from a greyscale image. Here, $f(p) = W - p$, where W is the pixel value representing white.

The most sophisticated pixel point processing is concerned with colour correction and alteration. We will not describe this fully until Chapter 6. Here, we will only consider the brightness and contrast alterations that are the typical applications of pixel point processing to greyscale images. Colour processing is an extension — although not a trivial one — of these greyscale adjustments. Once again, we will use Photoshop's tools to provide a concrete example, but any image editing software will offer the same functions, with roughly the same interface.

The crudest adjustments are made with the brightness and contrast sliders, which work like the corresponding controls on a monitor or television set. Brightness adjusts the value of each pixel up or down uniformly, so increasing the brightness makes every pixel lighter, decreasing it makes every pixel darker. Contrast is a little more subtle: it adjusts the range of values, either enhancing or reducing the difference between the lightest and darkest areas of the image. Increasing contrast makes the light areas very light and the dark areas very dark, decreasing it moves all values towards an intermediate grey. In terms of mapping functions, both of these adjustments produce a linear relationship that would be represented as a straight line on a graph: adjusting the brightness changes the intercept between the line and the y-axis; adjusting the contrast alters the gradient of the line.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 157
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

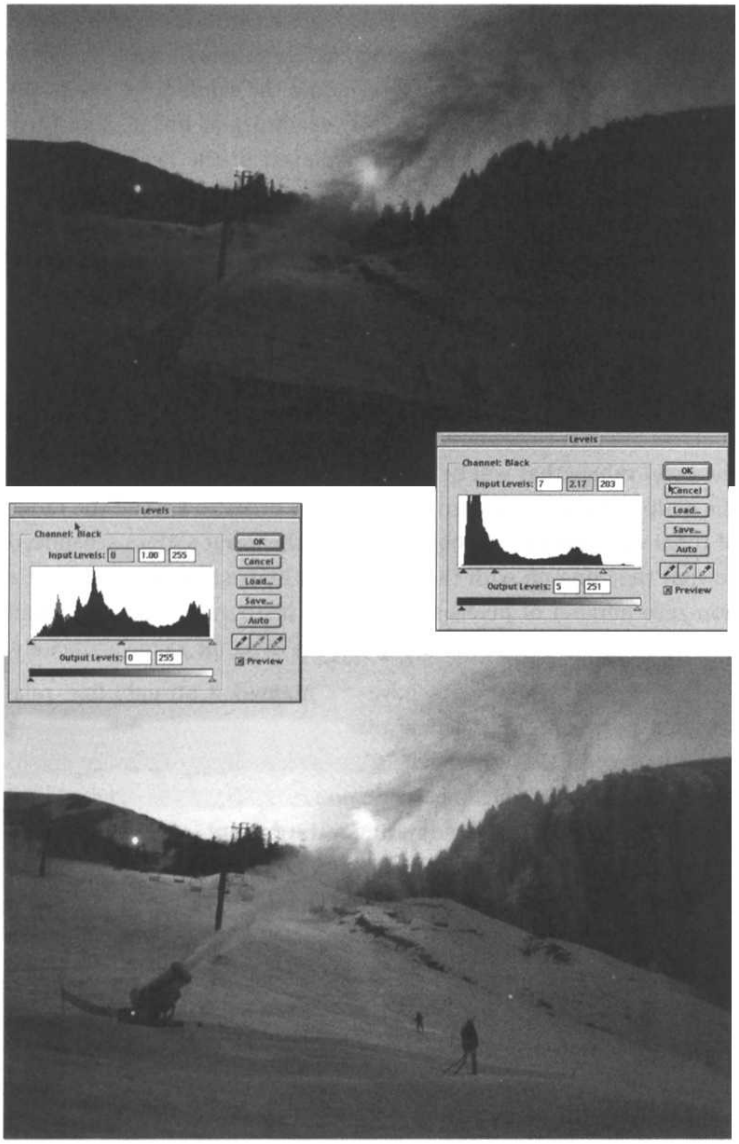


Figure 5.7
Level adjustments

More control over the shape of the mapping function is provided by the levels dialogue, which allows you to move the endpoints of a linear mapping function individually, thereby setting the white and black levels in the image. Graphically, these adjustments stretch or shrink the mapping function horizontally and vertically. To help with choosing suitable levels, a display called the *image histogram*

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 158
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

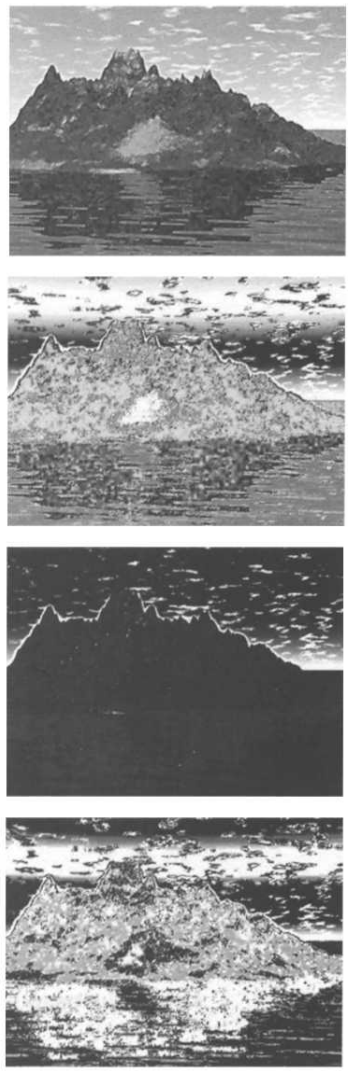


Figure 5.8
The effect of different curves on a single image

is used. This is a histogram showing the distribution of pixel values: the horizontal axis represents the possible values (from 0 to 255 in an 8-bit greyscale image), the bars show the number of pixels set to each value. Figure 5.7 shows two examples of images and their histograms. The histograms are displayed in the levels dialogue, with two sets of sliders below them, as shown. The upper set controls the range of input values. The slider at the left controls the pixel value that will be mapped to black, so, in graphical terms, it moves the intercept of the mapping function's line along the x-axis. The slider at the right controls the pixel value that is mapped to white, so it moves the top end of the line along the horizontal line corresponding to the maximum pixel value. The lower slider controls affect the output values in a similar way, i.e. they determine the pixel values that will be used for black and white, so they move the endpoints of the line up and down. In order to spread the range of tonal values evenly across the image, the input sliders are moved so that they line up with the lowest and highest values that have a non-zero number of pixels shown in the histogram. Moving beyond these points will compress or expand the dynamic range artificially.

So far, all the adjustments have maintained a straight-line relationship between old and new pixel values. The third slider that you can see on the upper levels control in Figure 5.7 allows you to produce a more flexible correspondence between original and modified pixel values, by adjusting a third point, corresponding to the mid-tones in the image. If an image's brightness is concentrated in a particular range, you can move the mid-point slider under the corresponding point on the histogram, so that the brightness values are adjusted to put this range in the centre of the available scale of values. Figure 5.7 shows the effect that level adjustments can achieve in bringing out detail that has been lost in an under-exposed photograph. The top image was shot at dusk, with too short an exposure; the levels dialogue below it shows the positions of the sliders that were used to produce the lower image. The changed image histogram can be seen in the lower levels dialogue box.

All of the brightness and contrast adjustment facilities described so far can be considered as making specialized alterations to the graph of the mapping function f to achieve particular commonly required adjustments to the values of individual pixels. In Photoshop, it is possible to take detailed control of this graph in the curves dialogue, where it can be reshaped by dragging control points, or completely redrawn with a pencil tool. The almost complete freedom to map

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 159
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

grey levels to new values that this provides permits some strange effects, but it also makes it easy to apply subtle corrections to incorrectly exposed photographs, or to compensate for improperly calibrated scanners.

Before any adjustments are made, the curve is a straight line with slope equal to one: the output and input are identical, f is an identity function. Arbitrary reshaping of the curve will cause artificial highlights and shadows. Figure 5.8 shows a single image with four different curves applied to it, to bring out quite different features. More restrained changes are used to perform tonal adjustments with much more control than the simple contrast and brightness sliders provide. For example, an S-shaped curve such as the one illustrated in Figure 5.9 is often used to increase the contrast of an image: the mid-point is fixed and the shadows are darkened by pulling down the quarter-point, while the highlights are lightened by pulling up the three-quarter-point. The gentle curvature means that, while the overall contrast is increased, the total tonal range is maintained and there are no abrupt changes in brightness.

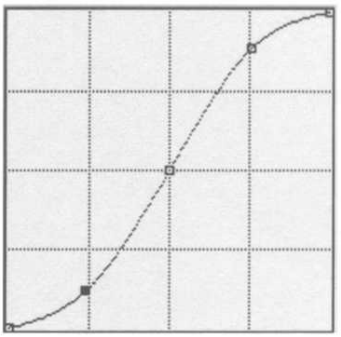


Figure 5.9
The S-curve for enhancing contrast

The adjustments we have just described compute a pixel's new value as a function of its old value. We can look at compositing as another form of pixel point processing, where a pixel's value is computed as a function of the values of two corresponding pixels in the images or layers being combined. That is, when we merge pixels p_1 and p_2 in two separate images, we compute a result pixel with value $p' = p_1 \oplus p_2$, where $\oplus$ is some operator. The different blending modes that are provided by image processing programs correspond to different choices of $\oplus$. Generally, non-linear operators are needed to perform useful merging operations. In particular, it is common to use a threshold function, of the form $p' = p_1$ if $p_1 > t$, $p' = p_2$, otherwise, where the threshold value t corresponds to the opacity setting for the layer.

Pixel Group Processing

A second class of processing transformations works by computing each pixel's new value as a function not just of its old value, but also of the values of neighbouring pixels. Functions of this sort perform *pixel group processing*, which produces qualitatively different effects from the pixel point processing operations we described in the preceding section. In terms of the concepts we introduced in Chapter 2, these operations remove or attenuate

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 160
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

certain spatial frequencies in an image. Such *filtering* operations can be implemented as operations that combine the value of a pixel with those of its neighbours, because the relative values of a pixel and its neighbours incorporate some information about the way the brightness or colour is changing in the region of that pixel. A suitably defined operation that combines pixel values alters these relationships, modifying the frequency make-up of the image. The mathematics behind this sort of processing is complicated, but the outcome is a family of operations with a simple structure.

It turns out that, instead of transforming our image to the frequency domain (for example, using a DCT) and performing a filtering operation by selecting a range of frequency components, we can perform the filtering in the spatial domain — that is, on the original image data — by computing a weighted average of the pixels and its neighbours. The weights applied to each pixel value determine the particular filtering operation, and thus the effect that is produced on the image's appearance. A particular filter can be specified in the form of a two-dimensional array of those weights. For example, if we were to apply a filter by taking the value of a pixel and all eight of its immediate neighbours, dividing them each by nine and adding them together to obtain the new value for the pixel, we could write the filter in the form:

$$\begin{matrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{matrix}$$

The array of weights is called a *convolution mask* and the set of pixels used in the computation is called the *convolution kernel* (because the equivalent of the multiplication operation that performs filtering in the frequency domain is an operation in the spatial domain called convolution).

Generally, if a pixel has coordinates (x, y), so that it has neighbours at $(x-1, y+1)$, $(x, y+1) \ldots (x, y-1)$, $(x+1, y-1)$, and we apply a filter with a convolution mask in the form:

$$\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix}$$

the value p' computed for the new pixel at (x, y) is

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 161
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

$$p' = ap_{x-1,y+1} + bp_{x,y+1} + cp_{x+1,y+1} + dp_{x-1,y} + ep_{x,y} + fp_{x+1,y} + gp_{x-1,y-1} + hp_{x,y-1} + ip_{x+1,y-1}$$

where $p_{x,y}$ is the value of the pixel at (x, y), and so on. The process is illustrated graphically in Figure 5.10.

Convolution is a computationally intensive process. As the formula just given shows, with a 3×3 convolution kernel, computing a new value for each pixel requires nine multiplications and eight additions. For a modestly sized image of 480×320 pixels, the total number of operations will therefore be 1382400 multiplications and 1228800 additions, i.e. over two and a half million operations. Convolution masks need not be only three pixels square — although they are usually square, with an odd number of pixels on each side — and the larger the mask, and hence the kernel, the more computation is required.[9]

This is all very well, but what are the visible effects of spatial filtering? Consider again the simple convolution mask comprising nine values equal to 1/9. If all nine pixels being convolved have the same value, let us say 117, then the filter has no effect: $117/9 \times 9 = 117$. That is, over regions of constant colour or brightness, this filter leaves pixels alone. However, suppose it is applied at a region including a sharp vertical edge. The convolution kernel might have the following values:

117 117 27
117 117 27
117 117 27

then the new value computed for the centre pixel will be 105. Moving further into the lighter region, to an area that looks like this:

117 27 27
117 27 27
117 27 27

gives a new pixel value of 57. So the hard edge from 117 to 27 has been replaced by a more gradual transition via the intermediate values 105 and 57. The effect is seen as a *blurring*. One way of thinking about what has happened is to imagine that the edges have

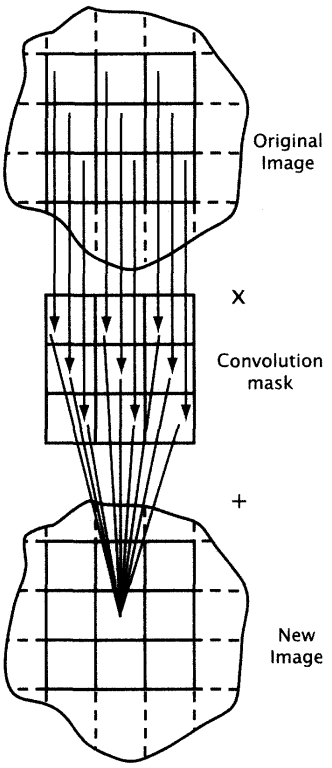


Figure 5.10
Pixel group processing with a convolution mask

9 Applying certain filters, particularly Gaussian blur (described below) to a large image file is often used to provide 'real world' benchmarking data for new computers.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 162
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

been softened by rubbing together the colour values of the pixels, in the same way as you blur edges in a pastel drawing by rubbing them with your finger. An alternative view, based on the concepts of signal processing, is that this operation produces a smoothing effect on the spatial waveform of the image, by filtering out high frequencies. (Engineers would refer to the operation as a *low pass filter.*)

Blurring is often used in retouching scans. It is useful for mitigating the effects of digital artefacts, such as the jagged edges produced by undersampling, Moiré patterns, and the blockiness resulting from excessive JPEG compression.

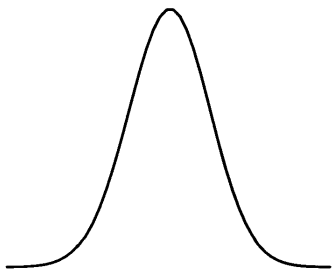


Figure 5.11
The Gaussian bell curve

Although the convolution mask we have just described is a classical blur filter, it produces a noticeably unnatural effect, because of the limited region over which it operates, and the all or nothing effect caused by the uniform coefficients. At the same time, the amount of blurring is small and fixed. A more generally useful alternative is *Gaussian blur*, where the coefficients fall off gradually from the centre of the mask, following the Gaussian 'bell curve' shown in Figure 5.11, to produce a blurring that similar to those found in nature. The extent of the blur — that is, the width of the bell curve, and hence the number of pixels included in the convolution calculation — can be controlled. Photoshop's dialogue allows the user to specify a 'radius' value, in pixels, for the filter. A radius of 0.1 pixels produces a very subtle effect; values between 0.2 and 0.8 pixels are good for removing aliasing artefacts. Higher values are used to produce a deliberate effect. A common application of this sort is the production of drop-shadows: an object is selected, copied onto its own layer, filled with black and displaced slightly to produce the shadow. A Gaussian blur with a radius between 4 and 12 pixels applied to the shadow softens its edges to produce a more realistic effect. A radius of 100 pixels or more blurs the entire image into incoherence; one of 250 pixels (the maximum) just averages all the pixels in the area the filter is applied to. Note that the radius specified is not in fact the limit of the blurring effect, but a parameter that specifies the shape of the bell curve — the blurring extends well beyond the radius, but its effect is more concentrated within it, with roughly 70% of the contribution to the value of the centre pixel coming from pixels within the radius.[10]

Figure 5.12 shows a typical application of Gaussian blur: a scanned watercolour painting has had a small blur, followed by a slight sharpening (see below), applied to remove scanning artefacts. The

10
The so-called radius is really the standard deviation of the normal distribution corresponding to the bell curve.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 163
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

result is an image that closely resembles the original, with the filters themselves indiscernible — the blurriness you see is the characteristic spreading of thin watercolour and has nothing to do with the filters. In contrast, the blur in Figure 5.13, with a radius of 29 pixels, transforms the image into something quite different. These pictures, and the remaining illustrations of filters in this section, are reproduced in colour in Plate 14, where the effects can be better appreciated.



Figure 5.12
A scanned image, corrected by filtering

Other types of blur are directional, and can be used to indicate motion. The final example in Plate 14, also shown in Figure 5.14, shows radial blur with the zoom option, which gives an effect that might suggest headlong flight towards the focal point of the zoom.

Blurring is a surprisingly useful effect when applied to digitized images — you might expect blur to be an undesirable feature of an image, but it conceals their characteristic imperfections; in the case of Gaussian blur, it does this in a visually natural way. Sometimes, though, we want to do the opposite, and enhance detail by sharpening the edges in an image. A convolution mask that is often used for this purpose is:

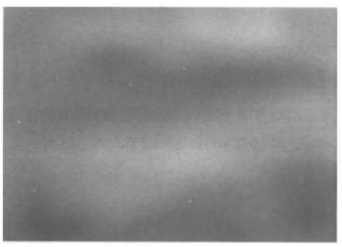


Figure 5.13
A large amount of Gaussian blur

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{matrix}$$

This mask filters out low frequency components, leaving the higher frequencies that are associated with discontinuities. Like the simple blurring filter that removed high frequencies, this one will have no effect over regions where the pixels all have the same value. In more intuitive terms, by subtracting the values of adjacent pixels, while multiplying the central value by a large coefficient, it eliminates any value that is common to the central pixel and its surroundings, so that it isolates details from their context.



Figure 5.14
Zooming radial blur

If we apply this mask to a convolution kernel where there is a gradual discontinuity, such as

$$\begin{matrix} 117 & 51 & 27 \\ 117 & 51 & 27 \\ 117 & 51 & 27 \end{matrix}$$

assuming that this occurs in a context where all the pixels to the left have the value 117 and those to the right 27, the new values computed for the three pixels on the central row will be 317, −75 and −45; since we cannot allow negative pixel values, the last two

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 164
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

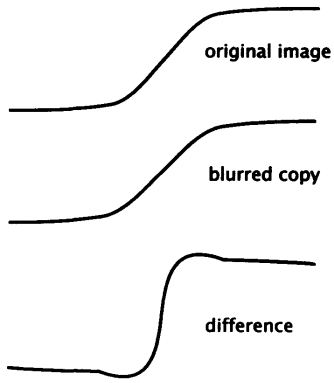


Figure 5.15
Unsharp masking

will be set to 0 (i.e. black). The gradual transition will have been replaced by a hard line, while the regions of constant value to either side will be left alone. A filter such as this will therefore enhance detail.

As you might guess from the example above, sharpening with a convolution mask produces harsh edges; it is more appropriate for analysing an image than for enhancing detail in a realistic way. For this task, it is more usual to use an *unsharp masking* operation. This is easiest to understand in terms of filtering operations: a blurring operation filters out high frequencies, so if we could take a blurred image away from its original, we would be left with only the frequencies that had been removed by the blurring — the ones that correspond to sharp edges. This isn't quite what we usually want to do: we would prefer to accentuate the edges, but retain the other parts of the image as well. Unsharp masking[11] is therefore performed by constructing a copy of the original image, applying a Gaussian blur to it, and then subtracting the pixel values in this blurred mask from the corresponding values in the original multiplied by a suitable scaling factor. As you can easily verify, using a scale factor of 2 leaves areas of constant value alone. In the region of a discontinuity, though, an enhancement occurs. This is shown graphically in Figure 5.15. The top curve shows the possible change of pixel values across an edge, from a region of low intensity on the left, to one of higher intensity on the right. (We have shown a continuous change, to bring out what is happening, but any real image will be made from discrete pixels, of course.) The middle curve illustrates the effect of applying a Gaussian blur: the transition is softened, with a gentler slope that extends further into the areas of constant value. At the bottom, we show (not to scale) the result of subtracting this curve from twice the original. The slope of the transition is steeper, and overshoots at the limits of the original edge, so visually, the contrast is increased. The net result is an enhancement of the edge, as illustrated in Figure 5.16, where an exaggerated amount of unsharp masking has been applied to the original image.

11
Originally a process of combining a photograph with its blurred negative.



Figure 5.16
Enhancing edges by unsharp masking



Figure 5.17
Unsharp masking applied after extreme Gaussian blur

The amount of blur applied to the mask can be controlled, since it is just a Gaussian blur, and this affects the extent of sharpening. It is common also to allow the user to specify a threshold; where the difference between the original pixel and the mask is less than the threshold value, no sharpening is performed. This prevents the operation from enhancing noise by sharpening the visible artefacts

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 165
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

it produces. (Notice how in Figure 5.16, the grain of the watercolour paper has been emphasized.)

Although sharpening operations enhance features of an image, it should be understood that they add no information to it. On the contrary, information is actually lost, although, if the sharpening is succesful, the lost information will be irrelevant or distracting. (It's more intuitively obvious that information is lost by blurring an image.) It should also be understood that although, in a sense, blurring and sharpening are opposites, they are not true inverses. That is, if you take an image, blur it and then sharpen it, or sharpen it and then blur it, you will not end up with the image you started with. The information that is lost when these operations are applied cannot be restored, although it is interesting to see features re-emerging in Figure 5.17 when Figure 5.13 is treated with an unsharp mask. This demonstrates how much information is actually preserved even under intense blurring.

Blurring and sharpening are central to the established scientific and military applications of image processing, but now that image manipulation software is also used for more creative purposes, some rather different effects are called for as well. Photoshop provides a bewildering variety of filters, and third party plug-ins add many more. Many of them are based on the type of pixel group processing we have described, with convolution masks chosen to produce effects that resemble different photographic processes or, with more or less success, the appearance of real art materials. These filters usually work by picking out edges or areas of the same colour, and modifying them; they are not too far removed from the more conventional blur and sharpen operations. Figure 5.18 shows the 'glowing edges' filter's effect on the seascape painting.

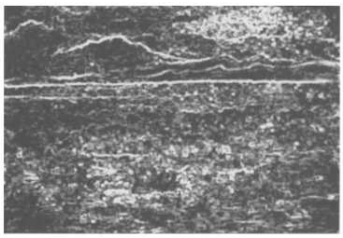


Figure 5.18
Glowing edges

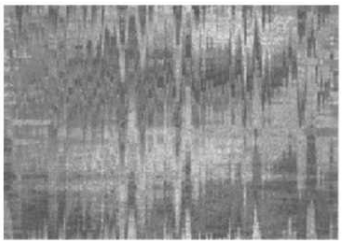


Figure 5.19
Distortion with a square wave



Figure 5.20
Twirled image

⇨ If you have access to Photoshop, you should investigate the `Custom` filter (on the `Other` sub-menu of the `Filter` menu). This allows you to construct your own convolution mask, by entering coefficients into a 5×5 matrix. The results are instructive and sometimes surprising.

Another group of filters is based on a different principle, that of moving selected pixels within an image. These produce various sorts of distortion. Figure 5.19 shows the seascape image modified by a square-wave pattern, while Figure 5.20 shows the twirl filter applied to its sharpened version from Figure 5.15, producing an entirely new image. As this example indicates, filters may be

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 166
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

combined. It is not untypical for designers with a taste for digital effects to combine many different filters in order to generate imagery that could not easily be made any other way.

Geometrical Transformations

Scaling, translation, reflection, rotation and shearing are collectively referred to as *geometrical transformations.* As we saw in Chapter 4, these transformations can be be applied to vector shapes in a very natural way, by simply moving the defining points according to geometry and then rendering the transformed model. Applying geometrical transformations to bitmapped images is less straightforward, since we have to transform every pixel, and this will often require the image to be resampled.

Nevertheless, the basic approach is a valid one: for each pixel in the image, apply the transformation using the same equations as we gave in Chapter 4 to obtain a new position in which to place that pixel in the transformed image. This suggests an algorithm which scans the original image and computes new positions for each of its pixels. An alternative, which will often be more successful is to compute the transformed image, by finding, for each pixel, a pixel in the original image. So instead of mapping the original's coordinate space to that of the transformed image, we compute the inverse mapping. The advantage of proceeding in this direction is that we compute all the pixel values we need and no more. However, both mappings run into problems because of the finite size of pixels.

For example, suppose we wish to scale up an image by a factor s. (For simplicity, we will assume we want to use the same factor in both the vertical and horizontal directions.) Thinking about the scaling operation on vector shapes, and choosing the inverse mapping, we might suppose that all that is required is to set the pixel at coordinates (x', y') in the enlarged image to the value at $(x, y) = (x'/s, y'/s)$ in the original. In general, though, x'/s and y'/s will not be integers and hence will not identify a pixel. Looking at this operation in the opposite direction, we might instead think about taking the value of the pixel at coordinates (x, y) in our original and mapping it to $(x', y') = (sx, sy)$ in the enlargement. Again, though, unless s is an integer, this new value will sometimes fall between pixels. Even if s is an integer only some of the pixels

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 167
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

in the new image will receive values. For example, if $s = 2$, only the even numbered pixels in even numbered rows of the image correspond to any pixel in the original under this mapping, leaving three quarters of the pixels in the enlarged image undefined. This emphasizes that, in constructing a scaled-up image we must use some interpolation method to compute values for some pixels.

However, it is not just in scaling up that interpolation is required. Whenever we appy a geometrical transformation to an image, it can send values into the gaps between pixels. Consider, for example, something as simple as moving a selected area of an image stored at 72 ppi one fifth of an inch to the right. Even scaling an image *down* can result in the same phenomenon unless the scale factor is a whole number. It should be clear from our discussion earlier in this chapter that changing the resolution of an image leads to similar problems.

A useful way of thinking about what is going on is to imagine that we are reconstructing a continuous image, so that we can find the required values in between the pixels of our sampled image, and then resampling it. Thus, the general problem is the same as the one we introduced when we discussed digitization in Chapter 2: how to reconstruct a signal from its samples. In practice, of course, we combine the reconstruction and resampling into a single operation, because we can only work with discrete representations.

We know that, for general images which may contain arbitrarily high frequencies because of sharp edges, the reconstruction cannot be done perfectly. We also know from sampling theory that the best possible reconstruction is not feasible. All we can hope to do is approximate the reconstruction to an acceptable degree of accuracy by using some method of *interpolation* to deduce the intervening values on the basis of the stored pixels. Several interpolation schemes are commonly employed; Photoshop provides three, for example, which we will describe next. As is usual in computing, the more elaborate and computationally expensive the algorithm used, the better the approximation that results.

Suppose that we are applying some geometrical transformation, and we calculate that the pixel at the point (x', y') in the resulting image should have the same value as some point (x, y) in the original, but x and y are not integers. We wish to sample the original image at (x, y), at the same resolution at which it is stored, so we can imagine drawing a pixel — call it the *target pixel* — centred at (x, y) which will be the sample we need. As Figure 5.21 shows, in

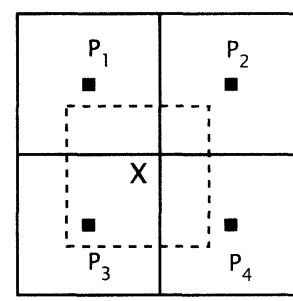


Figure 5.21
Pixel interpolation

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 168
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

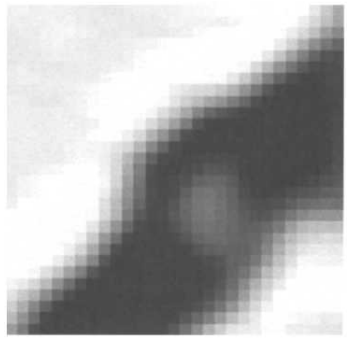


Figure 5.22
Nearest neighbour interpolation

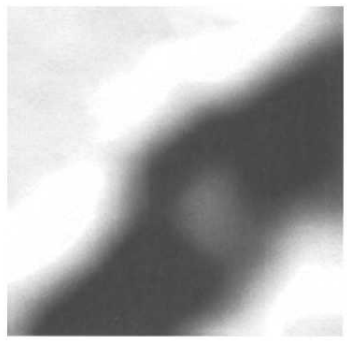


Figure 5.23
Bi-linear interpolation

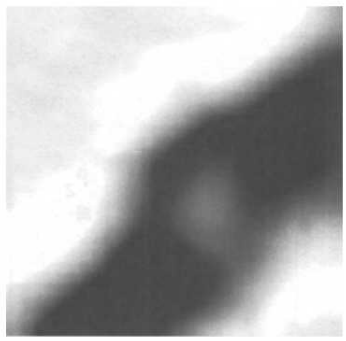


Figure 5.24
Bi-cubic interpolation

general this pixel may overlap four pixels in the original image. In the diagram, X marks the centre of the target pixel, which is shown dashed; P_1, P_2, P_3, and P_4 are the surrounding pixels, whose centres are marked by the small black squares.

The simplest interpolation scheme is to use the *nearest neighbour*, i.e. we use the value of the pixel whose centre is nearest to (x, y), in this case P_3. In general — most obviously in the case of upsampling or enlarging an image — the same pixel will be chosen as the nearest neighbour of several target pixels whose centres, although different, are close together enough to fall within the same pixel. As a result, the transformed image will show all the symptoms of undersampling, with visible blocks of pixels and jagged edges. An image enlarged using nearest neighbour interpolation will, in fact, look as if its original pixels had been blown up with a magnifying glass.

A better result is obtained by using *bi-linear interpolation*, which uses the values of all four adjacent pixels. They are combined in proportion to the area of their intersection with the target pixel. Thus, in Figure 5.21, the value of P_1 will be multiplied by the area enclosed by the dashed lines and the solid intersecting lines in the north-west quadrant, and added to the values of the other three pixels, multiplied by the corresponding areas.

If a and b are the fractional parts of x and y, respectively, then some simple mathematics shows that the value of the pixel at (x', y') in the result, whose target pixel is centred at (x, y) will be equal to

$$(1-a)(1-b)p_1 + a(1-b)p_2 + (1-a)bp_3 + abp_4$$

where p_i is the value of the pixel P_i, for $1 \le i \le 4$.

This simple area calculation is implicitly based on the assumption that the values change linearly in both directions (hence '*bi*-linearly') across the region of the four pixels. An alternative way of arriving at the same value is to imagine computing the values vertically above and below (x, y) by combining the values of the two pairs of pixels in proportion to their horizontal distances from x, and then combining those values in proportion to their vertical distances from y. In practice, the values are unlikely to vary in such a simple way, so that the bi-linearly interpolated values will exhibit discontinuities. To obtain a better result, *bi-cubic interpolation* can

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 169
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

be used instead. Here, the interpolation is based on cubic splines,[12] that is, the intermediate values are assumed to lie along a Bézier curve connecting the stored pixels, instead of a straight line. These are used for the same reason they are used for drawing curves: they join together smoothly. As a result, the resampled image is smooth as well.

Bi-cubic interpolation does take longer than the other two methods but relatively efficient algorithms have been developed and modern machines are sufficiently fast for this not to be a problem on single images. The only drawback of this interpolation method is that it can cause blurring of sharp edges.

Figures 5.22 to 5.24 show the same image enlarged using nearest neighbour, bi-linear, and bi-cubic interpolation.

12 We will not try to intimidate you with the equations for bi-cubic interpolation.

Further Information

[Bax94] is a good introduction to image manipulation. [NG96] describes many compression algorithms, including JPEG. [FvDFH96] includes a detailed account of re-sampling.

Exercises

1. Suppose you want to change the size of a bitmapped image, and its resolution. Will it make any difference which order you perform these operations in?

2. On page 126 we suggest representing the first 4128 pixels of the image in Figure 3.1 by a single count and value pair. Why is this not, in general, a sensible way to encode images? What would be a better way?

3. Our argument that no algorithm can achieve compression for all inputs rests on common sense. Produce a more formal proof, by considering the number of different inputs that can be stored in a file of N bytes.

4. We lied to you about Plate 4: the original painting was made on black paper (see Figure 5.25). In order to make it easier to reproduce, we replaced that background with the blue



Figure 5.25
The original iris scan

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 170
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

gradient you see in the colour plate. Describe as many ways as you can in which we might have done this.

5. Describe how you would convert a photograph into a vignette, such as the one shown in Plate 13, by adding an oval border that fades to white, in the fashion of late nineteenth century portrait photographers. How would you put an ornamental frame around it?

6. Explain why it is necessary to use an alpha channel for anti-aliased masks.

7. Compare and contrast the use of alpha channels and layer transparency for compositing.

8. Describe how the input-output curve of an image should be changed to produce the same effect as moving (a) the brightness, and (b) the contrast sliders. How would these adjustments affect the histogram of an image?

9. Describe the shape of the curve you would use to correct an image with too much contrast. Why would it be better than simply lowering the contrast with the contrast slider?

10. If asked to 'sharpen up' a scanned image, most experts would first apply a slight Gaussian blur before using the sharpen or unsharp mask filter. Why?

11. Motion blur is the smearing effect produced when a moving object is photographed using an insufficiently fast shutter speed. It is sometimes deliberately added to images to convey an impression of speed. Devise a convolution mask for a motion blur filter. How would you allow a user to alter the amount of motion blur? What other properties of the blurring should be alterable?

12. Why are the screen shots published in tutorial articles in computing magazines often hard to read?

13. Explain carefully why pixel interpolation may be required if a rotation is applied to a bitmapped image.

14. An alternative to using bi-linear or bi-cubic pixel interpolation when downsampling an image is to apply a low-pass filter (blur) first, and then use the nearest neighbour. Explain why this works.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 171
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Colour

6

Both vector graphics and bitmapped images can use colour. For most people, colour is such a commonplace experience that it is somehow surprising to discover that it is actually a rather complex phenomenon, in both its objective and subjective aspects. Representing colour in digital images and reproducing it accurately on output devices are consequently not at all straightforward.

Perhaps the most important thing to realize about colour is that you don't always need it. The existence of black and white photography and film demonstrates that people are perfectly well able to recognize and understand an image in the absence of colour — variations in brightness are quite adequate. Indeed, comparing the luminous elegance of a Fred Astaire film from the 1930s with the garish Technicolor of a 1950s MGM musical is enough to demonstrate that the addition of colour is not necessarily an improvement — a fact well understood by advertising agencies and the makers of music videos. Pragmatically, there are advantages to using images without colour. As we will see shortly, black and white bitmapped image files can be much smaller than coloured ones. Furthermore, black and white images are largely immune to the variations in colour reproduction of different monitors. You should not forget that some people do not have colour monitors, or prefer to use them in monochrome mode, and that a few people cannot see in colour at all, while many more cannot always distinguish between certain colours. By working in black and white (or, more accurately,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 172
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

shades of grey) you avoid producing images that may not be seen properly for these reasons.

But people have come to expect colour, and colour can add a great deal to an image, if it is used effectively. Sometimes colour is vital to the purpose for which an image is being used: for example, the colours in a clothing catalogue will influence people's buying decisions, and must be accurate to avoid disappointment and complaints.

The effective use of colour is not something that can be summarized in a few rules. The experience of artists and designers working in traditional media can be put to good use, and in large-scale multimedia production it is wise to ensure that there are artistically trained specialists available, and to leave colour decisions to them. As in most of the other areas we will describe, certain constraints associated with the digital nature of the images we can use in our multimedia productions will modify how we approach colour, so artistic sensibility must be augmented with some technical knowledge.

Colour and Science

Colour is a subjective sensation produced in the brain. In order to reproduce colour electronically, or manipulate it digitally, we need a model of colour which relates that subjective sensation to measurable and reproducible physical phenomena. This turns out to be a surprisingly difficult task to accomplish successfully.

Since light is a form of electromagnetic radiation, we can measure its wavelength — the wavelength of visible light lies roughly between 400nm and 700nm — and its intensity. We can combine these measurements into a *spectral power distribution (SPD)*, a description of how the intensity of light from some particular source varies with wavelength. Figure 6.1 shows the SPD of typical daylight. (Notice that it extends beyond the visible spectrum). In effect, an SPD is constructed by splitting light into its component wavelengths, much as a prism splits a beam of light into a spectrum when we repeat Isaac Newton's optics experiments in school, and measuring the intensity of each component.[1] Subjective experiments show that an SPD corresponds closely to what we mean by 'colour', in the sense that observers can successfully match light with a particular SPD to

1
In theory, an SPD ought to be a continuous function, but it is satisfactorily approximated by using samples at wavelengths separated by a suitable interval, for example 10nm, giving an SPD consisting of 31 components.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 173
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

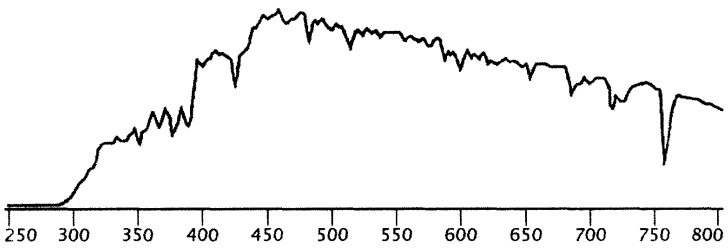


Figure 6.1
The spectral power distribution of daylight

a specific colour. However, SPDs are too cumbersome to work with when we are specifying colours for use in computer graphics, so we need to adopt a different approach.

You may have been told at some point in your education that the human eye contains two different sorts of receptor cells: *rods*, which provide night-vision and cannot distinguish colour, and *cones*, which in turn come in three different sorts, which respond to different wavelengths of light. The fact that our perception of colour derives from the eye's response to three different groups of wavelengths leads to the theory — called the *tristimulus* theory — that any colour can be specified by just three values, giving the weights of each of three components.

The tristimulus theory of colour is often summarized (inaccurately) by saying that each type of cone responds to one of red, green or blue light. It follows that the sensation of any colour can be produced by mixing together suitable amounts of red, green and blue light. We call red, green and blue the *additive primary colours*.[2] Putting that another way, we can define a particular colour by giving the proportions of red, green and blue light that it contains. It follows that we can construct television screens and computer monitors using pixels each made up of three dots of different types of phosphor, emitting red, green and blue light, and exciting them using three electron beams, one for each colour. To produce any desired colour we just have to adjust the intensity of each electron beam, and hence the intensity of the light emitted by the corresponding phosphors. Optical mixing of the light emitted by the three component dots of any pixel will make it look like a single pixel of the desired colour.

Since we *can* construct monitors like that, the simplified tristimulus theory is evidently more or less right, and much of the time we

2 These are not the artist's primary colours, which will be described later.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 174
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

can proceed on the basis of it. However, it *is* a simplified theory, and some subtle problems can arise if we ignore the more complex nature of colour. Fortunately, the worst of these are connected with printing, and do not often occur in multimedia work. Reproducing colour on a computer monitor, as we more often need to do, is more straightforward.

RGB Colour

The idea that colours can be constructed out of red, green and blue light leads to the *RGB colour model*, in which a colour is represented by three values, giving the proportions of red (R), green (G) and blue (B) light[3] which must be combined to make up light of the desired colour. The first question that arises is 'What do we mean by red, green and blue?'. The answer ought to be that these are the names of three colours corresponding to three standard primary SPDs, and in the television and video industries several such standards do exist. In computing, however, there is no universally accepted standard, although monitors are increasingly being built to use the primaries specified for High Definition TV (HDTV) by the ITU in its Recommendation ITU-R BT.709, and a standard version of RGB colour derived from this recommendation has been proposed. However, there is no real standard for red, green and blue on computer displays, and the colours produced in response to the any particular RGB value can vary wildly between monitors.

3
The primary blue is actually better described as 'blue-violet', and the red is an orangey shade.

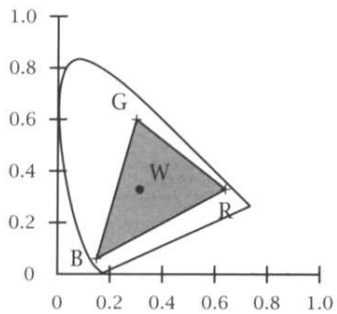


Figure 6.2
The RGB colour gamut

It is also important to be aware that it is *not* possible to represent any visible colour as a combination of red, green and blue components, however defined. Figure 6.2 shows a pictorial representation of the relationship between the colours that can be represented in that way — the so-called RGB *colour gamut* — and all the visible colours. The fin-shaped area is a spatial representation of all the possible colours. (For an explanation of exactly what is being plotted see the detailed comments below.) Shades of green lie towards the tip, with reds at the bottom right and blues at the bottom left. The triangular area of the RGB colour gamut is entirely enclosed within the fin, showing that there are some colours that cannot be produced by adding red, green and blue. It should be noted, though, that red, green and blue primaries do produce the largest gamut possible from simple addition of three primaries. Putting that another way, if you were to draw the largest triangle you

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 175
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

could fit inside the area of all possible colours, its vertices would correspond to colours you would call red, blue and green.

➲ Exactly what is being plotted in Figure 6.2? For a full explanation, you will have to consult the references given at the end of this chapter, but a rough outline is as follows. The diagram derives from work done under the auspices of the *Commission Internationale de l'Éclairage (CIE)* in 1931. In experiments in which people were asked to mix quantities of red, green and blue lights to produce a colour that matched a displayed sample, it was found that it was necessary to add negative amounts of primaries (by adding them to the displayed sample instead of the attempted match) in order to match some colours. This is now understood to be because the responses of the three type of cone cell are not simply used as colour stimuli, the way the voltages on the three electron guns of a monitor are, but are combined in a more complex way. The CIE defined a set of three primaries, known simply as X, Y and Z, whose spectral distributions matched the inferred spectral response of the eye of their 'standard observer', and could thus be used to produce any visible colour purely by addition. The Y component is essentially the brightness (more properly called *luminance* in this context). If we put

$$x = \frac{X}{X+Y+Z}, \; y = \frac{Y}{X+Y+Z} \text{ and } z = \frac{Z}{X+Y+Z}$$

you should be able to see that any colour can be fully specified by its x, y and Y values, since X and Z can be recovered from the equations. (Note that z is redundant.) Since Y is the luminance, x and y together specify the colour, independently of its brightness. It is x and y that label the axes in Figure 6.2. (This diagram is called the *CIE chromaticity diagram*, because it plots colour information, independent of brightness.) The curved area is obtained by plotting the value of an SPD comprising a single wavelength as its value varies from 400nm to 700nm. The area is closed by a straight line joining the extreme low and high frequencies — the 'line of purples', which corresponds to those colours which do not have a single identifiable wavelength as their dominant spectral component. The RGB gamut is contained inside a triangle with red, green and blue (as defined by the CIE) at its vertices.

Unfortunately, the primaries X, Y and Z, cannot be realized by physical light sources.

In practice, the majority of colours perceived in the world do fall within the RGB gamut, so the RGB model provides a useful, simple and efficient way of representing colours. A colour can

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 176
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

be represented by three values. We can write this representation in the form (r, g, b), where r, g and b are the amounts of red, green and blue light making up the colour. By 'amount', we mean the proportion of pure ('*saturated*') light of that primary. For example, if we express the proportions as percentages, (100%, 0%, 0%) represents pure saturated primary red, and (50%, 0%, 0%) a darker red, while (100%, 50%, 100%) represents a rather violent shade of mauve. Since black is an absence of light, its RGB colour value is (0%, 0%, 0%). White light is produced by mixing equal proportions of saturated light of all three primaries, so white's RGB colour value is (100%, 100%, 100%). We emphasize that the three values represent the amounts of *light* of the three primary colours which must be mixed to produce light of the specified colour. Do not confuse this *additive mixing* of colours with paint mixing, which is a *subtractive mixing* process, since paint absorbs light.[4] Computer monitors emit light, so any consideration of how colours are produced must be based on an additive model. Scanners work by detecting light reflected from the scanned document, so they too work with additive colours.

4 This is a simplification — see below.

The three numbers r, g and b are not absolute values in any sense, it is only their relative values that matter, so we can choose any convenient range, provided it allows us to distinguish enough different values. Like many important questions about colour, 'How many is enough?' can only be answered subjectively. Different cultures have different ideas about when two colours are different, and people differ in their ability to distinguish between different colours, but few, if any, can distinguish more than the nearly 16.8 million distinct combinations provided by using 256 different values for each of the red, green and blue component values.

Colour Depth

Of course, 256 is a very convenient number to use in a digital representation, since a single 8-bit byte can hold exactly that many different values, usually considered as numbers in the range 0 to 255. Thus, an RGB colour can be represented in three bytes, or 24 bits. The number of bits used to hold a colour value is often referred to as the *colour depth*: when three bytes are used, the colour depth is 24. We often also refer to *24 bit colour*, as a shorthand for colour with a 24 bit colour depth. In 24 bits, our mauve shade would be stored as (255, 127, 255), black is (0, 0, 0),

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 177
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

as always, and white is (255, 255, 255). (Remember that the values start at zero).

You may well suppose that 24 is not the only possible colour depth, and this is indeed the case. One other possibility, less common than it used to be, is 1-bit (*bi-level*) colour. A single bit allows us to distinguish two different colours. Usually these will be considered black and white, although the actual colours displayed will depend on the output device — orange and black monitors enjoyed a vogue for a while on ergonomic grounds, for example. A colour depth of 4 bits allows 16 different colours, which is clearly inadequate for displaying anything but the most simple of colour images; 16 different grey levels, on the other hand, can produce respectable greyscale images. Since greys are represented by RGB colour values (r, g, b), with $r = g = b$, a single value is sufficient to specify a grey level, the other two RGB components are redundant.

The 8 bit colour depth, where a single byte is used to hold a colour value, providing 256 colours, is in widespread use, both in monitors for domestic personal computers and on the World Wide Web. It affords several points of particular interest, and we will return to it shortly. Some computer systems use 16 bits to hold colour values. 16 is not divisible by 3, so when RGB values are stored in a 16 bit format, either one bit is left unused, or different numbers of bits are assigned to the three components. Typically, red and blue each use 5 bits, and green is allocated 6 bits, allowing twice as many different green values to be distinguished. This allocation of bits is based on the observation that the human eye is more sensitive to green light than to the other two primaries. (The cone cells in the eye are especially insensitive to blue, to compensate for the high levels of blue in daylight. The high sensitivity to green is presumably an evolutionary response to the preponderance of green in the environment in which human beings evolved.)

Although 24 bits are sufficient to represent more colours than the eye can distinguish, higher colour depths, such as 30, 36 or even 48 bits are increasingly being used, especially by scanners. Support for 48 bit colour is included in the specification of the PNG file format. These very large colour depths serve two purposes. Firstly, the additional information held in the extra bits makes it possible to use more accurate approximations when the image is reduced to a lower colour depth for display (just as images stored at high resolution will look better when displayed on a low resolution monitor than images stored at the monitor resolution). Secondly, it

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 178
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

is possible to make extremely fine distinctions between colours, so that effects such as chroma-key (see Chapter 10) can be applied very accurately.

The common colour depths are sometimes distinguished by the terms *millions of colours* (24 bit), *thousands of colours* (16 bit) and *256 colours* (8 bit), for obvious reasons (although 16 bits allows 65,536 values, so tens of thousands of colours would be a more accurate description). The names *true colour* and *hi colour* are sometimes used for 24 and 16 bit colour, respectively, but they are often used more loosely by marketing people. We will avoid these terms.

Colour depth is a crucial factor in determining the size of a bitmapped image: each logical pixel requires 24 bits for 24 bit colour, but only 8 for 8 bit, and just a single bit for 1 bit colour. Hence, if it possible to reduce the colour depth of an image from 24 bits to 8, the image file size will decrease by a factor of three (ignoring any fixed-size housekeeping information that is held in addition to the image data). Returning to an earlier point, an image made up of 256 different shades of grey — more than most people can distinguish — will be one third the size of the same image in millions of colours. Using an arbitrarily chosen set of 256 colours, on the other hand, is unlikely to produce an acceptable result. If the colour depth must be reduced, and grey-scale is not desired, then an alternative strategy must be employed.

Indexed Colour

So far, we have implicitly assumed that the stored R, G and B values are used directly to control the intensity of the monitor's three electron beams, thus determining the colour that is displayed. Often this is just what is done, especially in the case of 24 bit colour.[5] This arrangement is called *direct colour*. There is, however, an alternative, which is very widely used on low-end computer systems and the World Wide Web, known as *indexed colour*.

The screen of any colour monitor is capable of displaying the millions of colours that can be represented by 24-bit colour values, but it may well be the case that the video RAM (VRAM) provided in the monitor is not sufficient to hold a full-screen image at 3 bytes per pixel. (Table 6.1 shows the amount of VRAM needed to support each colour depth at a selection of common screen sizes.) Another

5 The relationship between the voltage applied to an electron beam and the intensity of the light emitted by a phosphor when it strikes it is non-linear, as is the response of the eye to the intensity of light which enters it, so the value is not simply used as a voltage, but the hardware is built to compensate somewhat for these non-linearities.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 179
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Table 6.1
Colour depth and VRAM

VRAM (Mbytes)	Screen Resolution	Max Colour Depth
1	640 × 480	16
	832 × 624	16
	1024 × 768	8
	1152 × 870	8
2	640 × 480	24
	832 × 624	24
	1024 × 768	16
	1152 × 870	16
4	640 × 480	24
	832 × 624	24
	1024 × 768	24
	1152 × 870	24
	1280 × 1024	16
8	640 × 480	24
	832 × 624	24
	1024 × 768	24
	1152 × 870	24
	1280 × 1024	24

constraint on colour depth is that an image file in 24-bit colour may be considered too large for the available disk space, or it may be felt to take too long to transmit over a network. If, for any of these reasons, we are constrained to use only one byte for each pixel, we can use at most 256 different colours in any one image. Using a standard set of 256 colours for all the images we might need, i.e. attempting to use 8-bit direct colour, is unacceptably restrictive. Indexed colour provides a means of associating a *palette* of 256 specific colours with each image. For example, if we wished to produce a pastiche of Picasso's 'blue period' paintings, we could use a palette holding shades of blue and obtain a reasonable result. If we had had to use only the blues from a set of 256 colours spread evenly over the spectrum, all of the subtlety and possibly much of the content would be lost.

One way of thinking about indexed colour is as a digital equivalent of painting by numbers. In a painting by numbers kit, areas on the painting are labelled with small numbers, which identify pots of paint of a particular colour. When we use indexed colour, pixels store a small number that identifies a 24-bit colour from the palette associated with the image. Just as each painting by numbers outfit includes only those paints that are needed to colour in one picture,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 180
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

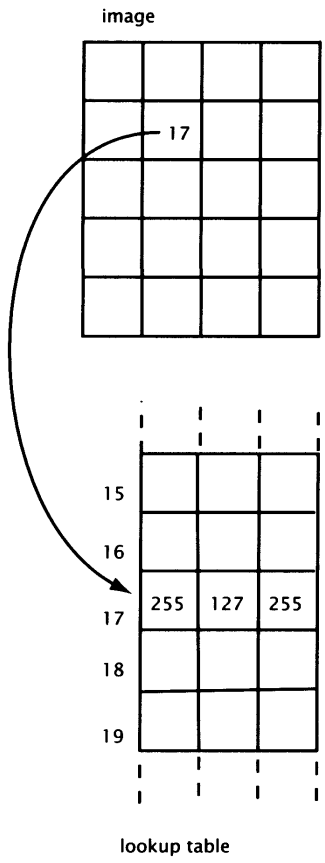


Figure 6.3
Using a colour lookup table

so the palette includes only the 24-bit RGB values for the colours used in the image. When the image is displayed, the graphics system looks up the colour from the palette corresponding to each single-byte value stored at each pixel, and uses the value it finds as the colour of that pixel.

Experienced programmers will immediately recognize the strategy: it is an example of the folk axiom that all problems can be solved by adding a level of indirection. Hence, instead of trying to use eight bits to hold an RGB value, we use it to hold an index into a table, with 256 entries, each of which holds a full 24-bit RGB colour value. For example, if a particular pixel was to be displayed as the colour whose 24-bit RGB value was (255, 127, 255), instead of storing that value in the image, we could store an index identifying the table location in which it was stored. Supposing (255, 127, 255) was held in the 18th entry of the table, the pixel would hold the offset 17 (see Figure 6.3). Such an indexed table of colour values is called a *Colour Lookup Table (CLUT)* or, more simply, a *palette.* You may find it helpful to consider the CLUT as carrying out a mapping from *logical colours*, stored for logical pixels, to *physical colours*, displayed by physical pixels. The looking up operation that performs this mapping is usually carried out by the monitor's video hardware; all the displaying program needs to do is load the correct palette. (Even if the display hardware supports 24-bit direct colour, indexed colour may sometimes be used to reduce the size of image files; in that case, the CLUT lookup may be done by software.) If no palette is supplied, a default system palette will be used (the system palettes on the major platforms are different).

It may help you to understand what is going on when indexed colour is being used if you consider what happens if a program displays two images in different windows at the same time, and those images use different palettes. Unless the program has been designed to operate in an unusual way, the colour palette associated with the active window will be loaded. Say this palette is mostly made up of shades of blue, and the 5th entry is a pale sky-blue. Then every pixel with a value of 4 (palette offsets start at 0) will be coloured that pale sky-blue — including pixels in the other image, if its window is still visible. This other image may be made up of shades of brown, with the 5th element of its palette a reddish-brown; nevertheless, while the blue image's window is active, the pixels that should be reddish-brown will be pale sky-blue, since, although the palette is associated with the image, it is applied to the entire screen. Hence, if the brown

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 181
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

image becomes active, the pale sky-blues in the first window will turn reddish-brown. A logical colour value does not identify a colour absolutely, only relative to a palette. You will appreciate that there is a problem if two images are to be displayed in the same window, as they must be if they both occur on the same Web page. It will be necessary to use a single palette to display both. This may require the number of colours in each individual image to be restricted to an even smaller number, unless the colour make-up of both images is the same.

If indexed colour is to be used an image file needs to store the palette along with the actual image data, but the image data itself can be smaller by a factor of three, since it only needs eight bits per pixel. For any reasonably sized image, there will be a net saving of space. This saving may be increased if the file format allows palettes with fewer than 256 entries to be stored with images that have fewer colours, and uses the minimum number of bits required to index the palette for each colour value.

Of the file formats mentioned in Chapter 3, PNG, BMP, TGA and TIFF allow the use of a colour palette; GIF requires it as it only supports 8-bit indexed colour images; an SGI image file may have a palette associated with it in a separate file. JFIF and SPIFF files do not support indexed colour: JPEG images stored in 24-bit colour must be reduced to 8-bit if necessary at the time they are displayed. PostScript and its derivatives, such as EPS and PDF, provide sophisticated support for indexed colour, although its use is usually confined to bitmapped images embedded in the PostScript, since its use with vector shapes, where the colour specification is a small part of the image model, is less compelling.

For created images, a palette of 256 colours will often be tolerable and, since you have control over the colours you use, you can work within the restriction, though subtle variations will be lost. However, with photographic or scanned material, or images that already exist in 24-bit colour, it will be necessary to cut down the number of colours when you prepare a version that uses 8-bit indexed colours. What can be done with the areas of the image that should be displayed in some colour which is not in the reduced palette?

Obviously, the missing colour must be replaced by one of those that is in the palette. There are two popular ways of doing this. The first is to replace the colour value of each individual pixel with the CLUT index of the nearest colour. This can have undesirable

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 182
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

effects: not only may the colours be distorted, but detail may be lost when two similar colours are replaced by the same one, and banding and other visible artefacts may appear where gradations of colour are replaced by sharp boundaries. (These effects are collectively known as *posterization*; posterization is sometimes used deliberately as an effect, for example, to simulate cheaply printed late 1960s underground posters.)

➲ What do we mean by 'nearest' colour? A simple-minded view is that the colour nearest to some value (r,g,b) is the one with RGB value (r',g',b') such that $\sqrt{(r'-r)^2+(g'-g)^2+(b'-b)^2}$ (the Euclidean distance between the two colour values) is minimized. This computation does not take account of the non-linearity of our perceptions of colour differences, so some correction should be applied to the given formula, although computer graphics programs often fail to do so.

When posterization is unacceptable, the replacement of colours missing from the palette can be done another way. Areas of a single colour are replaced by a pattern of dots of several different colours, in such a way that optical mixing in the eye produces the effect of a colour which is not really present. For example, say our image includes an area of pale pink, but the palette that we must use does not include that colour. We can attempt to simulate it by colouring some of the pixels within that area red and some white. This process is called *dithering*.[6] It is done by grouping together pixels in the area to be coloured, and applying colour to individual pixels within each group in a suitable pattern. The process is an extension of the use of half-toning, used to print greyscale images on presses that can only produce dots of pure black or white. At low resolutions, dithering may produce poor results, so it is better suited to high resolution work.

6 You may sometimes see the term 'dithering' used incorrectly just to mean reducing the number of colours.

Figure 6.4 shows five distinct ways of distributing black and white among a 2×2 block of pixels. If the pixels were small enough and the image was viewed from a sufficient distance, these patterns would be interpreted as five different shades of grey (including black and white) — see Figure 6.5, where we have exaggerated the size of the pixels to show how the half-tones are built up. In general, a block of $n \times n$ pixels can simulate $n^2 + 1$ different grey levels, so you can see that as the size of the block over which we dither is increased, so is the number of grey levels, but at the expense of resolution.

If each pixel can be one of 256 different colours, instead of just black or white, then the corresponding patterns can simulate

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 183
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

millions of colours. However, these simulated colours are, in effect, being applied to pixels four times the area of those actually on the screen, hence the effective resolution of the image is being halved, resulting in a loss of detail. While this is usually acceptable for printing, where resolutions in excess of 600dpi are common, it is often intrusive when images are being displayed on 72dpi monitors. Other artefacts may also be generated when the patterns are superimposed; these can be minimized by clever choice of dot patterns for the generated colours, although this has the effect of cutting down the number of different colours that can be simulated.

This leaves the question of which colours should be used in a palette. Ideally, you will fill the palette with the most important colours in your image. Often, these will be the most common, in which case it is easy for a program like Photoshop to construct the palette automatically by examining the original 24-bit version when it converts it to indexed colour. Sometimes, though, the use of colour will be more complex, and it may be necessary to construct the palette by hand (or, more likely, edit an automatically constructed palette) to ensure that all the vital colours are present.

You cannot guarantee that every program that displays your image will necessarily use your palette — some may just fall back on the default system palette, with dismaying results. If you have reason to believe that that is a possibility, it is usually better to convert to a system palette yourself when preparing the image, and make any adjustments you can to retrieve some quality instead of letting the system do its own colour conversion. Unfortunately, the system palettes for the major platforms are different. A restricted set of 216 colours, usually referred to as the *Web-safe palette*, is the only palette you can rely on to be reproduced by Web browsers on any system using 8-bit colour, and if you must know which colours are going to be displayed on any platform the Web-safe palette is your best option.

Plate 15 summarizes this section. The left-hand column shows, at the top, a 24-bit colour image, with, below it, the same image using a custom palette of its 256 most common colours; the palette is shown at the bottom. There is little discernible difference between the two images. The top picture in the middle column shows the same image again, this time using the Web-safe palette, with dithering applied. Again, and perhaps surprisingly, there is little discernible difference, but, if you look closely, you can see that the dithering has introduced some slight pixelation. Below this, is



Figure 6.4
Dithering patterns

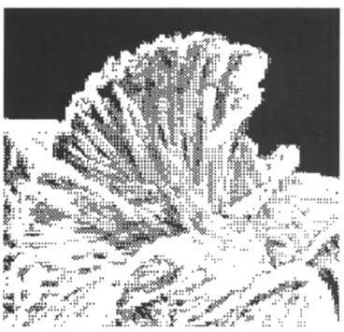


Figure 6.5
Grey produced by dithering

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 184
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

the same image, using the same palette, but without dithering, now badly posterized; the Web-safe palette is shown below. In the final column, the top image is a greyscale version, showing how much detail is retained despite the absence of colour. The lower image is indexed colour again, this time using the palette shown below it, which was constructed from the most common colours in the image that is reproduced as Plate 17.

Other Colour Models

The RGB colour model is the most important means of representing colours used in images for multimedia, because it corresponds to the way in which colour is produced on computer monitors, and it is also how colour is detected by scanners. Several other colour models are in use, and you should know something about them because you may be required to work with images that have already been prepared for some other medium, especially print, or to use software which is based on a different view of colour from RGB.

CMYK

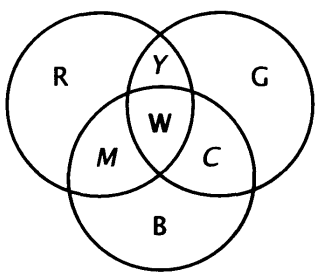


Figure 6.6
Complementary colours

You have probably seen at some time the experiment, originally devised by Thomas Young in 1801, illustrated in Plate 16. Beams of the three additive primaries red, green and blue are shone onto a white surface, which reflects all of the light falling on it, so that they overlap. Where only one colour lands, we see that colour; where all three overlap, we see white, as we would expect from the previous section. Where two colours overlap, we see a new colour formed by the mixing of the two additive primaries. These new colours are a pale shade of blue, usually called *cyan*, a slightly blueish red, called *magenta*, and a shade of *yellow* (see Figure 6.6). Each of these three is formed by adding two of the additive primaries. Since all three additive primaries combine to form white light, we could equally say that cyan, for example, which is the mixture of blue and green, is produced by subtracting the remaining primary, red, from white light.

We can express this effect pseudo-algebraically. Writing R, G and B for red, green and blue, C, M and Y for cyan, magenta and yellow,[7] and W for white, and using + to mean additive mixing of light,

7
The Y that stands for yellow is not to be confused with the Y that stands for luminance elsewhere in this chapter.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 185
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

and – to mean subtraction of light, we have

$$\begin{aligned} C &= G + B = W - R \\ M &= R + B = W - G \\ Y &= R + G = W - B \end{aligned}$$

In each equation, the colour on the left is called the *complementary colour* of the one at the extreme right; for example, magenta is the complementary colour of green.

The relevance of this experiment is two-fold. Firstly, it is the basis for a theory of colour aesthetics which has had a great influence on the use of colour in art and design. Secondly, the idea of forming colours by subtraction of light instead of addition provides a colour model appropriate to ink and paint, since these are substances which owe their coloured appearance to the way they absorb light.

An important point to grasp is that the light that is reflected from a coloured surface is not changed in colour by the process of reflection. For example, when you hold a glossy photograph under a bright white light, the light that bounces off it will produce white reflections that interfere with the colour of the photograph itself. The dyes on a surface such as paper do not supply colour to light reflected off the surface, but to light that penetrates through them and gets reflected or scattered back from beneath it. During the light's journey through the particles of dye, ink or paint, the pigments absorb light at some frequencies. The light that emerges thus appears to be coloured. When paints are mixed or dyes are overlaid, the combination absorbs all the frequencies absorbed by the individual components. When, for example, we talk of 'cyan ink', we mean ink that, when it is applied to white paper and illuminated by white light will absorb the red component, allowing the green and blue, which combine to produce the cyan colour, to be reflected back (see Figure 6.7). If we apply a layer of such an ink to white paper, and then add a layer of magenta, the magenta ink will absorb incident green light, so the combination of the cyan and magenta inks produces a blue colour (the additive primary blue, that is), as shown in Figure 6.8. Similarly, combining cyan and yellow inks produces green, while magenta combined with yellow gives red. A combination of all three colours will absorb all incident light, producing black. Mixtures containing different proportions of cyan, magenta and yellow ink will absorb red, green and blue light in corresponding proportions, thus (in theory) producing the same range of colours as the addition of red, green and blue

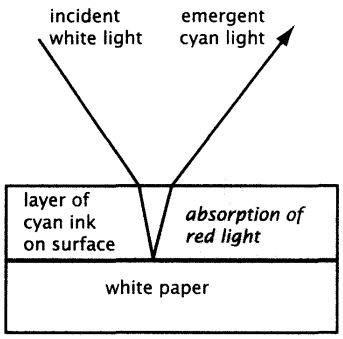


Figure 6.7
Effect of coloured ink

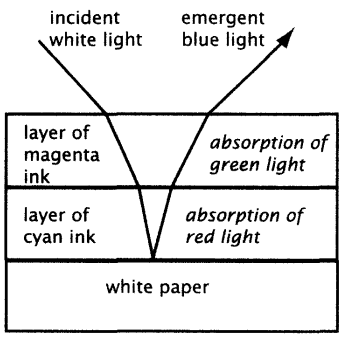


Figure 6.8
Mixing coloured inks

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 186
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

primary lights. Cyan, magenta and yellow are called the *subtractive primaries* for this reason. These subtractive primaries are the primary colours which the artist working in conventional media must use.

In practice, it is not possible to manufacture inks which absorb only light of precisely the complementary colour. Inevitably, some unwanted colours are absorbed at the same time. As a result, the gamut of colours that can be printed using cyan, magenta and yellow is not the same as the RGB gamut. Furthermore, combining actual inks of all three colours does not produce a very good black. On top of this, applying three different inks is not very good for your paper and leads to longer drying times. For these reasons, in magazine and book printing, the three subtractive primaries are augmented with black. The four colours cyan, magenta, yellow and black, when used in printing, are called *process colours*, and identified by their initials CMYK. Figure 6.9 shows the CMYK colour gamut and compares it with the RGB gamut displayed earlier. You will see that the CMYK gamut is smaller than RGB, but is not a strict subset — there are CMYK colours that fall outside the RGB gamut, and vice versa. You will also see that, because of the different colour mixing method and the presence of the black, the gamut is not triangular.

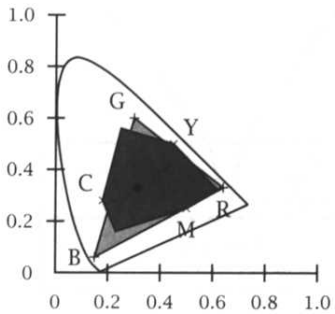


Figure 6.9
The CMYK colour gamut

For printing, an understanding of the subtractive CMYK model of colour, and also of the properties of inks and papers is vital to high quality colour reproduction. For multimedia, CMYK is of less direct relevance. If it should happen that you wish users of your multimedia production to be able to print copies of the images it contains, in colour, then you should ensure that they only contain colours within the CMYK gamut, otherwise there will be colour shifts on the printed version. If you are incorporating images that have previously been used in some print medium, you may find they are stored in a format in which colours are specified as C, M, Y and K values, instead of R, G and B. When the images are converted to a suitable file format for multimedia, it may be necessary also to convert the CMYK colour values into RGB. (Programs such as Photoshop can do this.) Having done so, you may be able to apply some colour transformations to improve the appearance by taking advantage of the larger RGB gamut.

Finally, if you come from a printing or painting background, or any pre-digital art and design discipline except photography, film or video, you may find it more natural to specify colours using

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 187
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

CMYK, since the subtractive mixing of CMYK colours is more like the mixing of inks and paints — the primaries are the same, for example. You should also realize that painting programs which simulate natural media will simulate the subtractive colour mixing that comes from overlaying different coloured paint. Plate 17 shows the effect of painting with thin layers of yellow, magenta and cyan using the simulated paints of Painter. The red, green, and blue, and the other colours are all formed by overlaying thin layers of these primaries with a simulated airbrush; the mixing is subtractive.

⇨ Many people's thinking about primary colours is confused by the habit school teachers have of telling children that the primary colours are red, yellow and blue, referring to the cheerful poster-paint primaries familiar from the classroom, not the artists' primary pigments, which are given different names by different manufacturers, but whose appearance corresponds well to the subtractive primaries magenta, yellow and cyan. The confusion stems from the breadth with which we apply the names of different hues: mixing any yellow with any blue will produce something we would call green, for example. Of the colours available in the sort of paints commonly found in junior schools, red, yellow and blue provide the closest approximations to the primaries. They can be mixed to give a range of colours, but not the full range obtainable when true primary pigments are available. (Purples are especially disappointing.)

HSV

Breaking a colour down into its component primaries makes sense from a theoretical point of view, and reflects the way monitors, scanners and colour printers work, but as a way of describing colour, it does not correspond to the way in which we experience colours in the world. Looking at a pale (cyan) blue, you probably don't relate it to a mixture of green and blue-violet light, but rather to other blues, according to how nearly pure blue it is, or how much of a green-ish or purple-ish blue it appears, and to how light or dark it is.

In more formal terms, we can consider a colour's *hue*, its *saturation*, and its *brightness*. In physicists's terms, hue is the particular wavelength at which most of the energy of the light is concentrated (the *dominant wavelength*); if you like, hue is the pure colour of light. In less scientific terms, we usually identify hues by names. Famously, Isaac Newton identified seven hues in the spectrum, the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 188
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

familiar red, orange, green, blue, indigo and violet of the rainbow. Newton had his own mystical reasons for wanting seven colours in the rainbow and it is more normal to distinguish just four: red, yellow, green and blue, and define hue informally as the extent to which a colour resembles one, or a mixture of two, of these.

A pure hue can be more or less 'diluted' by mixing it with white: the dominant hue (wavelength) remains the same, but the presence of other hues makes the colour paler. A colour's saturation is a measure of its purity. Saturated colours are pure hues; as white is mixed in, the saturation decreases. A colour's appearance will be modified by the intensity of the light: less light makes it appear darker. The brightness of a colour is a measure of how light or dark it is.

⇨ The equation of hue with the measurable quantity dominant wavelength is appealing, but not quite as simple as one would hope. Some hues cannot be identified with a single wavelength. Most people would consider purple to be a hue, but the SPD for purple has peaks at both short and long wavelengths (red and blue). We know, though, from the discussion of subtractive mixing, that we could consider purple to be made by subtracting light at the dominant wavelength of a shade of green from white light. In other words, the hue purple is associated with a dominant wavelength, but the dominance takes the form of a negative peak. This notion is, perhaps, no more incongruous than the idea that a hole is where an electron isn't.

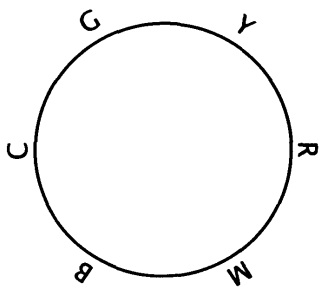


Figure 6.10
A colour wheel

In terms of paint, a hue is a pure colour. Adding white decreases its saturation, producing a tint. Adding black decreases its brightness, producing a tone.

Since the early nineteenth century, painters and other visual artists who like to systematize their thinking about colours have organized them into a 'colour wheel': the different hues are organized in a circle, with the subtractive primaries equally spaced around the perimeter, and the additive primaries in between them so that each primary is opposite its complement (see Figure 6.10). In art terms, it is the subtractive primaries which are usually referred to as primary colours. Their complements are called secondary colours, and the colours produced by mixing primaries and secondaries are tertiary colours. (The tertiary colours are sometimes added to the colour wheel in between the primaries and secondaries.)

This colour wheel can be extended to produce an alternative colour model. First, the distinct primary, secondary and tertiary hues can

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 189
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

be augmented to produce a continuum extending all the way round a circle. Any hue can be specified as an angle around the circle, usually measured counter-clockwise, relative to red at 0°. Saturation can be added to the model by putting white at the centre of the circle, and then showing a gradation of tints from the saturated hues on the perimeter to the pure white at the centre. To add brightness to this model, we need a third dimension. Imagine the colour disk just described as being on top of a cylinder, made up of similar slices whose brightness decreases as we move downwards, so that, instead of white at the centre we have increasingly dark greys, until at the bottom, there is black. On each slice, the colours will be correspondingly toned down. (See plate 18, which shows three such slices.) A particular colour can be identified by its hue H (how far round the circle from red it is), its saturation S (how near the central axis) and its brightness value V (how far up the cylinder it is), hence this form of colour description is called the HSV model.

⇨ As we move towards the ends of the HSV cylinder, the variability in saturation will be smaller. At the bottom, there is less light, so less scope for adding white, while at the top, there is more light, and less scope for adding black. Accordingly, a better model is actually provided by a double cone, rather than a cylinder, which narrows towards a single black point at the bottom, and a single white point at the top. The colour model based on this geometry is called *HLS*, which stands for hue, saturation, and lightness.

You may sometimes see references to the *HSB* model. This is identical to the HSV model just described, using the B to stand for brightness instead of the less obvious V for value.

Industrial designers use a different approach to colour specification, based on swatches of standard colours that can be reproduced accurately. The Pantone system is widely used for this purpose, and has been adapted for incorporation into computer graphics applications.

Although the HSV model superficially resembles one of the ways in which painters think about colour, it should be remembered that the arrangement of hues around the circle is really no more than a visual mnemonic for colour harmony — it is otherwise arbitrary and has no physical basis.

All colour models provide a way of specifying a colour as a set of numbers, and of visualizing their relationships in a three-dimensional space. This way of describing colour is independent of the many subjective factors which normally play a part in

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 190
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

our perception of colour. For computer programs, a numerical, objective, colour specification is necessary. For people, it is less so, and most people find the numerical specification of colours non-intuitive. Within graphics programs, though, it is normal to choose colours interactively using a *colour picker* dialogue. These may be based on any of the colour models we have discussed, and allow you to choose colours by manipulating controls that affect the values of the components, while seeing a sample of the resulting colour displayed. Figure 6.11 shows the RGB and HSV colour pickers provided by the Macintosh system software. In the HSV colour picker, a slider is used to control the V value, while the corresponding slices through the cylinder are displayed alongside, and a colour is selected by clicking on a point in the disk. The RGB picker simply uses sliders to control the three components.

Figure 6.11 also shows the witty 'crayon picker', which can be used to choose a colour from the system palette. More conventional palette pickers just present swatches of the available colours.

Colour Spaces Based on Colour Differences

For some purposes it is useful to separate the brightness information of an image from its colour. Historically, an important motivation for doing this was to produce a means of transmitting colour television signals that would be compatible with older black and white receivers. By separating brightness and colour, it is possible to transmit a picture in such a way that the colour information is undetected by a black and white receiver, which can treat the brightness as a monochrome signal. Once a means of separating colour and brightness had been devised, it became possible to process the two signals separately in other ways. In particular, it is common practice in analogue broadcasting to use less bandwidth to transmit the colour than the brightness, since the eye is less sensitive to colour than to brightness. There is evidence to suggest that the human visual apparatus processes brightness and colour separately, too, so this form of representation also holds some physiological interest.

The basis of the RGB representation of colour is that light can always be broken down into three components. What we consider as brightness — the extent to which a pixel appears to be lighter or darker — is clearly related to the values of the red, green and blue

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 191
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

components of the colour, since these tell us how much light of each primary is being emitted. It would be tempting to suppose that the brightness could be modelled simply as $(R + G + B)/3$, but this formula is not quite adequate, because it takes no account of the eye's differing sensitivity to the different primaries. Green contributes far more to the perceived brightness of a colour than red, and blue contributes hardly at all. Hence to produce a measure of brightness from an RGB value, we must weight the three components separately. There are several different formulae in use for performing this computation; that recommended for modern cathode ray tubes, as used in most computer monitors and TV sets is:

$$Y = 0.2125R + 0.7154G + 0.0721B$$

The quantity Y defined here is called *luminance*.

You should be able to see that red, green and blue values can be reconstructed from luminance and any two of the primaries. It is normal, though, for technical reasons, to use some variant of a representation consisting of Y together with two *colour difference* values, usually $B - Y$ and $R - Y$. We say 'some variant' because different applications require that the three components be manipulated in different ways. For analogue television, a non-linearly scaled version of Y, properly denoted Y', is used together with a pair of weighted colour difference values, denoted U and V. Out of carelessness, the term *YUV colour* is often used to refer to any colour space consisting of a brightness component together with two colour difference components. Digital television uses a variant called $Y'C_BC_R$, which is like Y'UV but uses different weights for the colour difference computation.

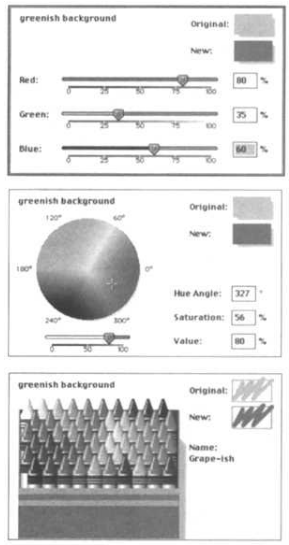


Figure 6.11
Colour pickers

Device-independent Colour Spaces

RGB and CMYK are the most commonly used colour models, but they are both *device-dependent*: different monitors provide different red, green and blue primaries; different sorts of ink and paper produce different cyan, magenta, yellow and even black. Given an RGB value such as $(255, 127, 255)$, you don't know exactly what colour is going to be produced on any particular monitor, and you have no guarantee that it will be the same on any two monitors, or even on the same monitor at different times.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 192
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

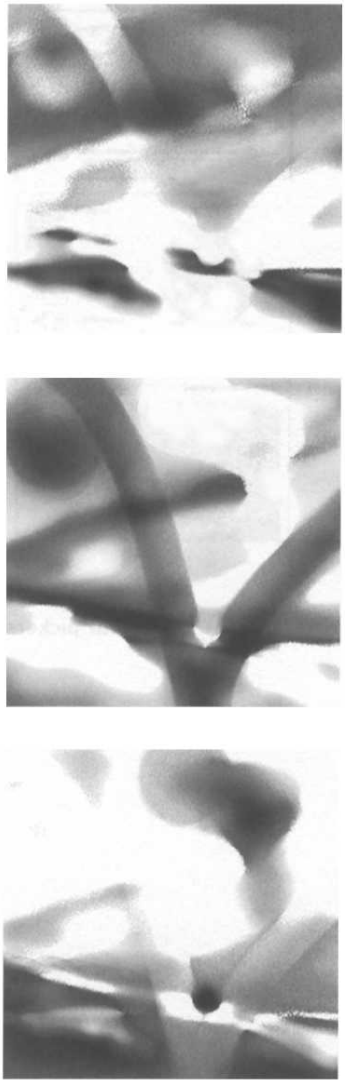


Figure 6.12
Red, green and blue channels of Plate 17

As we mentioned in the technical note on page 157, the *Commission Internationale de l'Éclairage (CIE)* has carried out work to produce an objective, *device-independent* definition of colours. Its basic model, the *CIE XYZ colour space*, uses three components X, Y and Z to approximate the three stimuli to which the colour-sensitive parts of our eyes respond. This model is device-independent, but as well as being awkward to work with and impossible to realize with physical light sources, it suffers (as do the RGB, CMYK, HSV and most other colour models) from not being *perceptually uniform*. A perceptually uniform model would be one in which the same change in one of the values produced the same change in appearance, no matter what the original value was. If, for example, RGB was a perceptually uniform model then changing the R component from 1 to 11 would produce the same increase in perceived redness as changing it from 101 to 111. However, it doesn't. This means that computations of the distance between two colours based on their RGB values do not reflect what we consider the degree of difference between them, which in turn makes finding a closest colour match when reducing an image to a restricted palette difficult.

⇨ You will be familiar with perceptual non-uniformity if you have ever owned a cheap audio amplifier: very small movements of the volume control at low levels make the sound much louder or quieter, while much larger movements at the top end make little discernible difference to how loud the sound seems.

For many years, the CIE tried to produce a colour model that was perceptually uniform, but never quite succeeded. Two models, refined from the original XYZ model to be much more nearly perceptually uniform were produced in 1976. These are the L*a*b* and L*u*v* models (often written simply Lab and Luv, or CIELAB and CIELUV).

In both models, the L* component is a uniform luminance and the other two components are colour differences. In L*a*b*, they specify colour in a way that combines subtractively, as in CMYK. 'Lab colour' is widely used in the pre-press industry as a standard specification of colour difference values for printing. L*u*v* is a similar model, but the colour difference values work additively, as in RGB, so L*u*v* provides a device-independent model of colour suitable for monitors and scanners.

These models do not correspond to any easily understandable way of thinking about colour, so you are unlikely to use them directly.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 193
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

However, they do provide a basis for the device-independent treatment of colour, and are used as a reference point for systems that try to guarantee that the colours we specify in our images will look as nearly as possible the same on whatever output device is being used.

Channels and Colour Correction

Earlier, we implied that in a 24-bit RGB colour image, a single 24-bit value is stored for each pixel. While this is the way colours are stored in some file formats, there is another possibility: for each pixel, three separate 8-bit values, one for each of red, green and blue, are stored. That is, in terms of data structures, instead of storing an image in a single array of 24-bit values, we store it in three arrays, each consisting of single bytes. This representation has some computational advantages, since 3-byte quantities are awkward to manipulate on most machines. It also provides a useful way of organizing colour images conceptually, irrespective of whether the physical storage layout is actually of this form.

Each of the three arrays can itself be considered as an image. When we consider each colour separately, it only makes sense to consider these as greyscale images. Each of the three greyscale images making up the colour image is called a *channel*, so we speak of the red channel, or the green or blue channel. Figure 6.12 shows the three channels of the image in Plate 17.

Each channel can be manipulated separately: since it is an image, the manipulations can be done using the usual image editing tools provided by Photoshop and similar programs. In particular, levels and curves can be used to alter the brightness and contrast of each channel independently. When the altered channels are recombined, the colour balance of the composite image will have changed. Colour correction operations performed in this way are frequently required to compensate for the deficiencies of scanners and other input devices. Plate 19 shows an example: the scanned photograph on the left has a marked colour cast; in the middle image, this has been corrected; the image on the right shows how colour adjustment can be taken further to produce an artifically coloured image.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 194
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

The previous paragraph may make adjusting the levels of each channel sound like a simple operation. In practice, although some colour adjustments are straightforward — maximizing the contrast in all three channels, for example — producing a desired result can be very time-consuming, calling for considerable experience and fine judgement. Some image manipulation programs come equipped with 'wizards' or 'assistants', software modules with some degree of artificial intelligence, to help in some common situations calling for colour correction.

Sometimes, though, a simpler approach is adequate. Just as the brightness and contrast controls encapsulate a class of tonal adjustments, so the *colour balance* and *hue and saturation* adjustments provide a less refined interface to the adjustment of levels in the three colour channels. Colour balance provides three sliders, one for each of the pairs of complements cyan and red, magenta and green, and yellow and blue, which allow you to adjust the relative proportions of each pair. The adjustment can be applied to shadows, midtones, or highlights separately. The hue and saturation adjustment performs a similar operation, by allowing you to vary each of the H, S, and V components of colours within specified ranges: reds, greens, blues, and so on. This provides a more intuitive way of adjusting the colours if you are used to working with HSV specifications.

A different way of making colour changes which, like the two sets of controls just described, works across all the channels at once, is to replace a specified colour wherever it appears with a different one. This is like doing a search and replace operation on a text file using an editor, except that the target and replacement are colours not text strings. Photoshop's version of this operation is somewhat more sophisticated: you can set a tolerance value, which causes pixels within a certain percentage of the target value to be altered, and use H, S, and V sliders to specify a change to those components of pixels that match the target, instead of just specifying an exact replacement colour. In effect, this command creates a mask, in a similar manner to the magic wand described in Chapter 5 and then makes colour adjustments to selected areas of the image.

Although we are only discussing RGB colour, images stored in any colour space can be separated into channels. In particular, the channels of a CMYK image correspond to the colour separations needed by commercial printing processes, while the channels of a $Y'C_BC_R$ image correspond to the components of a video signal.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 195
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Furthermore, the idea that a greyscale image can be considered to be one channel of a composite image is extended in the concept of alpha channel, which we introduced in Chapter 5. Although alpha channels behave completely differently from the colour channels that make up the displayed image, their common representation has some useful consequences.

One of the most common applications of this duality is in inserting people into a separately photographed scene, often one where it is impossible for them to be in reality. This can be done by photographing the person against a blue background (ensuring that none of their clothing is blue). All of the image data for the person will be in the red and green channels; the blue channel will be an image of the area not occupied by the person. An alpha channel constructed as a copy of the blue channel (an operation built in to any software that manipulates image channels) will therefore mask out the background, isolating the figure. It is then a simple matter to copy the figure, using the mask to define it as the selection, and paste it into, for example, an underwater scene. This *blue screen* technique is commonly used in video and film-making, as well as in constructing fake images.

Other sorts of colour image processing can be implemented by applying some greyscale processing to each of the channels separately. An important example is JPEG compression, which, although it is most commmonly applied to colour images, is defined in the standard to operate on 8-bit quantities, so that each channel is compressed individually. One advantage of this is that the compression algorithm is unaffected by the colour space used for the image, it just takes whatever channels are there and compresses them. This offers the freedom to choose a colour space, and apply pre-processing to the channels. It is common for JPEG compressors to transform an image into $Y'C_BC_R$, and then downsample the colour difference components, since the eye is less sensitive to colour variation than to brightness variations. This same 'chroma downsampling' step is applied to video signals, both for compression and transmission, as we will see in Chapter 10.

Consistent Colour

Colour adjustment is messy, and getting it wrong can cause irreparable damage to an image. It would be much better to get

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 196
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

things right first time, but the varying colour characteristics of different monitors and scanners make this difficult. Some recent developments in software are aimed at compensating for these differences. They are based on the use of *profiles*, describing how devices detect and reproduce colour.

We don't need much information to give a reasonable description of the colour properties of any particular monitor. We need to know exactly which colours the red, green and blue phosphors emit (the R, G and B *chromaticities*). These can be measured using a suitable scientific instrument, and then expressed in terms of one of the CIE device-independent colour spaces. We also need to know the maximum saturation each component is capable of, i.e. we need to know what happens when each electron beam is full on. We can deduce this if we can characterize the make-up and intensity of white, since this tells us what the (24-bit) RGB value (255, 255, 255) corresponds to. Again, the value of white — the monitor's *white point* — can be specified in a device-independent colour space.

⇨ You will sometimes see white point specified as a *colour temperature*, in degrees absolute. This form of specification is based on the observation that the spectral make-up of light emitted by a perfect radiator (a so-called 'black body') depends only on its temperature, so a black body temperature provides a concise description of an SPD. Most colour monitors for computers use a white point designed to correspond to a colour temperature of 9300°K. This is far higher than daylight (around 7500°K), or a conventional television monitor (around 6500°K), in order to generate the high light intensity required for a device that will normally be viewed under office lighting conditions. (Televisions are designed on the assumption that they will be watched in dimly lit rooms.) The 'white' light emitted by a monitor when all its three colours are at full intensity is actually quite blue.

Computer monitors are not actually black bodies, and so their SPDs deviate from the shape of the black body radiation, which means that colour temperature is only an approximation of the characteristic of the white point, which is better specified using CIE colour values.

Finally, the most complex element in the monitor's behaviour is the relationship between the RGB values presented to it by the graphics controller, and the intensity of the light emitted in response. This relationship is not a simple linear one: the intensity of light emitted in response to an input of 100 is not ten times that produced by an input of 10, which in turn is not ten times that produced by an input

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 197
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

of 1. Physical devices do not work as conveniently as that. However, the response can be captured to some extent using a single number, referred to as the display's γ (often written out as *gamma*).

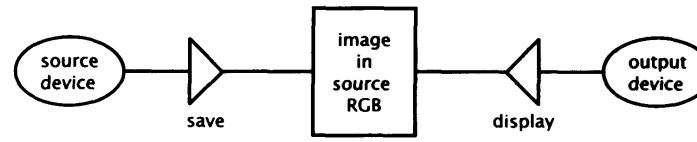
➲ Why γ? Because the *transfer characteristic* of a display — the relationship between the light intensity emitted by the screen (I) and the voltage applied to the electron gun (V) is often modelled by the transfer function $I = V^{\gamma}$, where γ is a constant. Unfortunately, this model is not entirely accurate, and one of the sources of variability between monitors lies in the use of incorrect values for γ which attempt to compensate for errors in the formula. Another is the fact that some display controllers attempt to compensate for the non-linearity by adjusting values according to an inverse transfer function before they are applied to the electron guns, while others do not.

Armed with device-independent values for red, green and blue, the white point and γ, it is possible to translate any RGB colour value into an absolute, device-independent, colour value in a CIE colour space that exactly describes the colour produced by the monitor in response to that RGB value. This is the principle behind the practice of *colour management*.

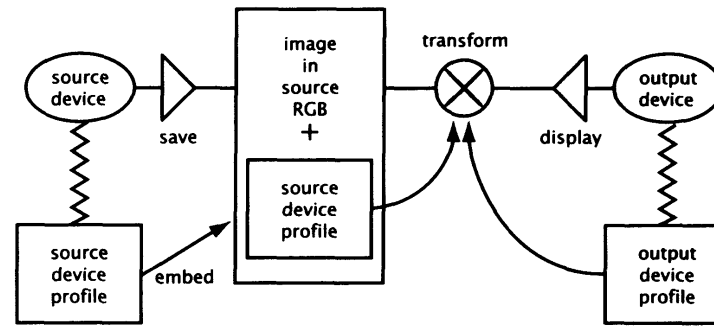
Figure 6.13(a) shows a typical situation calling for colour management: an image is prepared using some input device, which might be a monitor used as the display by a graphics editor, or a scanner. In either case, the image will be stored in a file, using RGB values which reflect the way the input device maps colours to colour values -- its *colour profile*. We say that the image is stored using the input device's colour space. Later, the same image may be displayed on a different monitor. Now the RGB values stored in the image file are mapped by the output device, which probably has a different profile from the input device. The colours, stored in the colour space of the input device are interpreted as if they were in the colour space of the output device. In other words, they come out wrong.

One way of correcting this is shown in Figure 6.13(b). Information from the input device's profile is embedded in the image file. EPS, TIFF, JFIF and PNG files are able to accommodate such information, in varying degrees of detail. At the least, the R, G and B chromaticities, white point and γ can be included in the file. Software on the machine being used to display the image can, in principle, use this information to map the RGB values it finds in the image file to colour values in a device-independent colour space. Then, using the device profile of the output monitor, map those

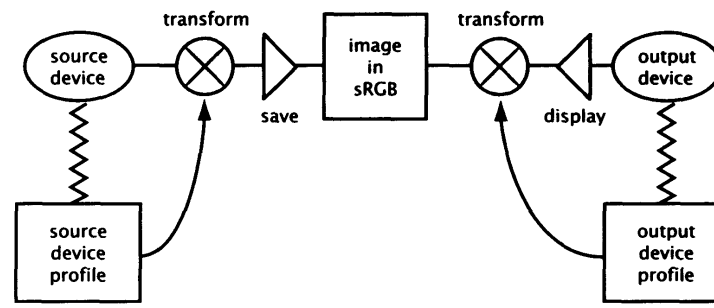
DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 198
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046



(a) No CMS, no standard colour space



(b) CMS



(c) Using the sRGB colour space

Figure 6.13
Colour management

values to the colour space of the output device, so that the colours are displayed exactly as they were on the input device. In practice, it is more likely that the two profiles would be combined and used to map from the input device colour space to the output device colour space directly. It is, of course, possible, that some of the colours in the input device colour space are not available in the output device colour space, so colour management can only actually guarantee that the colours will be displayed exactly as they were

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 199
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

intended within the capabilities of the output device. If software uses colour management consistently, the approximations made to accommodate a restricted colour gamut will be the same, and the output's colour will be predictable. If the display software has not been written to take advantage of device profiles, the display will be no better than it would be without colour management.

An alternative way of using colour management software is to modify the colours displayed on a monitor using the profile of a chosen output device. In this way, colour can be accurately previewed. This mode of working is especially useful in pre-press work, where actually producing printed output may be expensive or time-consuming.

To obtain really accurate colour reproduction across a range of devices, device profiles need to provide more information than simply the RGB chromaticities, white point and a single figure for γ. In practice, for example, the γs for the three different colours are not necessarily the same. As already stated, the actual transfer characteristics are not really correctly represented by γ; a more accurate representation is needed. If, as well as display on a monitor, we also wished to be able to manage colour reproduction on printers, where it is necessary to take account of a host of issues, including the C, M, Y and K chromaticities of the inks, spreading characteristics of the ink, and absorbency and reflectiveness of the paper, even more information would be required — and different information still for printing to film, or video.

Since the original impetus for colour management software came from the pre-press and printing industries, colour management has already been developed to accommodate these requirements. The *International Colour Consortium (ICC)* has defined a standard device profile, which supports extremely elaborate descriptions of the colour characteristics of a wide range of devices. ICC device profiles are used by colour management software such as Apple's ColorSync, and the Kodak Precision Color Management System, to provide colour management services. ColorSync provides colour management at a system software level, making it easy for individual applications to incorporate extremely sophisticated colour management services. Manufacturers of scanners, monitors and printers routinely produce ICC profiles of their devices. TIFF and EPS files can accommodate complete ICC profiles.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 200
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

⇨ Colour management is not much use unless accurate profiles are available. Unfortunately, no two devices are exactly identical. In the case of monitors, the situation is even worse, since tubes age and their colour characteristics change over time. Although a generic profile produced by the manufacturer for one line of monitors is helpful, to take full advantage of colour management it is necessary to calibrate individual devices, at relatively frequent intervals (once a month is often advised). Some high end monitors are able to calibrate themselves automatically. For others, it is necessary to use a special measuring device in conjunction with software that displays a sequence of colour values and, on the basis of the measured output of the screen, generates an accurate profile.

You may wonder why the profile data is embedded in the file. Why is it not used at the input end to map the colour values to a device-independent form, which can then be mapped to the output colour space when the image is displayed? The work is split between the two ends and no extra data has to be added to the file (see Figure 6.13(c)). The reason is that most existing software does not work with device-independent colour values, so it could not display the images at all. If it ignores a device profile, things are no worse than they would have been if it was not there.

Clearly, though, it would be desirable to use a device-independent colour space for stored colour values. The sRGB (standard RGB) colour model attempts to provide such a space. As you can guess, it is based on the RGB model, and specifies standard values for the R, G and B chromaticities, white point, and γ. The standard values have been chosen to be typical of the values found on most monitors.[8] As a result, if the display software is not aware of the sRGB colour space and simply displays the image without any colour transformation, it should still look reasonable. Use of sRGB colour is especially suitable for graphics on the World Wide Web, because there most images are only ever destined to be displayed on a terminal. A standard colour space based on RGB is of no help in maintaining consistency of colours for printing.

Having read this section, you may well ask 'Why bother?' It may not seem like the end of the world if the colours in your image are a bit off when it is displayed on somebody else's monitor. Consider, for example, the use of Web pages as online shopping catalogues. For many products, the colour is important. (To take an extreme example, think about buying paint over the Internet.) One of the factors driving the development of colour management

8 Or so its proponents claim. Other experts believe the chosen values are not the best. Photoshop 5.0 used sRGB as its default colour space; in response to users' reactions, this default was almost immediately changed in the 5.01 update.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 201
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

and its incorporation into Web browsers is the desire to facilitate online shopping. As well as the development of the sRGB colour space, this has led to the development of browser plug-ins providing full colour management facilities based on ColorSync. If online shopping seems crassly commercial, consider instead the use of the World Wide Web and CD-ROMs by art galleries to deliver catalogues. Here again the best possible colour reproduction is vital if the catalogues are to serve their function properly.

Further Information

[JMF94] surveys many aspects of colour; [Got98] considers colour in digital media from a designer's point of view. [Poy96] includes technical material on colour, especially in the context of video. [FvDFH96] also has a somewhat technical description of colour spaces and the physics of colour. [SACM] is the document that defines the sRGB colour space.

Exercises

1. Comment on the advisability of using translucent tangerine coloured plastics in the surround of a computer monitor.

2. If you consider the three components of an RGB colour to be cartesian coordinates in a 3-dimensional space, and normalize them to lie between 0 and 1, you can visualize *RGB colour space* as a unit cube, as shown in Figure 6.14. What colours correspond to the eight corners of this cube? What does the straight line running from the origin to $(1,1,1)$ shown in the figure represent? Comment on the usefulness of this representation as a means of visualizing colour.

3. Why do RGB colour values (r,g,b), with $r = g = b$ represent shades of grey?

4. Why do you think we chose to show the CIE chromaticity diagram in black and white as Figure 6.2, instead of providing a colour plate showing the actual colours?

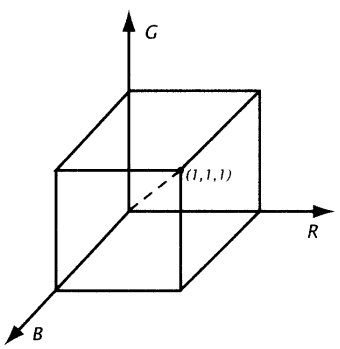


Figure 6.14
The RGB colour space

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 202
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

5. If a CLUT has 256 entries (so pixels only need a single byte), each 3 bytes wide, how large must an image be before using indexed colour provides a net saving of space?

6. For which of the following images would the use of indexed colour be satisfactory?

   (a) A reproduction of your national flag.

   (b) A photograph of the surface of Mars.

   (c) A photograph of yourself on the beach.

   (d) A still from a black and white Fred Astaire movie.

   In which cases is the system palette likely to be acceptable without dithering?

7. Give an example of an image whose most important colour is not one of its most commonly occurring colours. When might this cause a problem?

8. If you gave a child a paintbox containing red, green and blue paints only, will they be able to mix an adequate range of colours to paint the ceiling of the Sistine chapel?

9. Which colour space should be used for storing images and why?

10. Suppose you had an image with too much red in it, perhaps because of poor lighting. Explain what adjustments you would make to its red channel to compensate. What undesirable side-effect would your adjustment have, and what would you do about it?

11. Superman's uniform is blue, with a yellow S on the chest, and a red cape. How would you make an image of Superman flying, without recourse to his superpowers?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 203
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Characters and Fonts

7

Text has a dual nature: it is a visual representation of language, and a graphic element in its own right. Text in digital form must also be a representation of language; that is, we need to relate bit patterns stored in a computer's memory or transmitted over a network to the symbols of a written language. When we consider the display of stored text, its visual aspect becomes relevant. We then become concerned with such issues as the precise shape of characters, their spacing and the layout of lines, paragraphs and larger divisions of text on the screen or page. These issues of display are traditionally the concern of the art of *typography*. Much of the accumulated typographical practice of the last several centuries can be adapted to the display of the textual elements of multimedia.

In this chapter, we consider how the fundamental units of written languages — characters — can be represented in a digital form, and how the digital representation of characters can be turned into a visual representation for display. We will show how digital font technology makes it possible to approximate the typographical richness of printed text in the textual components of multimedia.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 204
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Character Sets

In keeping with text's dual nature, it is convenient to distinguish between the lexical *content* of a piece of text and its *appearance*. By content we mean the characters that make up the words and other units, such as punctuation or mathematical symbols. (At this stage we are not considering 'content' in the sense of the meaning or message contained in the text.) The appearance of the text comprises its visual attributes, such as the precise shape of the characters, their size, and the way the content is arranged on the page or screen. For example, the content of the following two sentences[1] is identical, their appearance is not:

1 From the short story *Jeeves and the Impending Doom* by P.G. Wodehouse.

```
The Right Hon was a tubby little chap who looked as if
he had been poured into his clothes and had forgotten
to say 'When!'
```

The Right Hon was a tubby little chap who looked as if he had been *poured* into his clothes and had forgotten to say 'When!'

We all readily understand that the first symbol in each version of this sentence is a capital T, even though one is several times as large as the other, has some additional strokes, and is darker. To express their fundamental identity, we distinguish between an *abstract character* and its graphic representations, of which there is a potentially infinite number. Here, we have two graphic representations of the same abstract character.

As a slight over-simplification, we could say that the content is the part of a text that carries its meaning or semantics, while the appearance is a surface attribute that may affect how easy the text is to read, or how pleasant it is to look at, but does not substantially alter its meaning. In the example just given, the fixed-width, typewriter-like font of the first version clearly differs from the more formal book font used for most of the second, but this and the gratuitous initial dropped cap and use of different fonts do not alter the joke. Note, however, that the italicization of the word 'poured' in the second version, although we would normally consider it an aspect of the appearance like the use of the sans serif font for 'Right Hon', implies an emphasis on the word that is missing in the plain version (and also in the original story). Hence, the distinction between appearance and content is not quite as clear-cut as one might think. Nevertheless, it is a useful distinction, because

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 205
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

it permits a separation of concerns between these two qualities that text possesses.

Abstract characters are grouped into alphabets. Each particular alphabet forms the basis of the written form of a certain language or group of languages. We consider any set of distinct symbols to be an alphabet.[2] This includes the set of symbols used in an ideographic writing system, such as those used for Chinese and Japanese, where each character represents a whole word or concept, as well as the phonetic letters of Western-style alphabets, and the intermediate syllabic alphabets, such as Korean Hangul. In contrast to colloquial usage, we include punctuation marks, numerals, and mathematical symbols in an alphabet, and treat upper- and lower-case versions of the same letter as different symbols. Thus, for our purposes, the English alphabet includes the letters A, B, C,... Z, and a, b, c,... z, but also punctuation marks, such as comma and exclamation mark, the digits 0, 1,..., 9, and common symbols such as + and =.

2 We do not define 'symbol'. In the abstract, an alphabet can be any set at all, but in practical terms, the only symbols of interest will be those used for writing down some language.

To represent text digitally, it is necessary to define a mapping between (abstract) characters in some alphabet and values that can be stored in a computer system. The only values that we can store are bit patterns. As we explained in Chapter 2, these can be interpreted as integers to base 2, so the problem becomes one of mapping characters to integers. As an abstract problem, this is trivial: *any* mapping will do, provided it associates each character of interest with exactly one number. Such an association is called, with little respect for mathematical usage, a *character set*; its domain (the alphabet for which the mapping is defined) is called the *character repertoire*. For each character in the repertoire, the character set defines a *code value* in its range, which is sometimes called the set of *code points*. The character repertoire for a character set intended for written English text would include the twenty-six letters of the alphabet in both upper- and lower-case forms, as well as the ten digits and the usual collection of punctuation marks. The character repertoire for a character set intended for Russian would include the letters of the Cyrillic alphabet. Both of these character sets could use the same set of code points; provided it was not necessary to use both character sets simultaneously (for example, in a bilingual document) a character in the English alphabet could have the same code value as one in the Cyrillic alphabet. The character repertoire for a character set intended for the Japanese Kanji alphabet must contain at least the 1945 ideograms for common use and 166 for names sanctioned by the Japanese Ministry of

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 206
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Education, and could contain over 6000 characters; consequently, it requires far more distinct code points than an English or Cyrillic character set.

The mere existence of a character set is adequate to support operations such as editing and searching of text, since it allows us to store characters as their code values, and to compare two characters for equality by comparing the corresponding integers; it only requires some means of input and output. In simple terms, this means that it is necessary to arrange that when a key is pressed on a keyboard, or the equivalent operation is performed on some other input device, a command is transmitted to the computer, causing the bit pattern corresponding to the character for that key to be passed to the program currently receiving input. Conversely, when a value is transmitted to a monitor or other output device, a representation of the corresponding character should appear.

There are advantages to using a character set with some structure to it, instead of a completely arbitrary assignment of numbers to abstract characters. In particular, it is useful to use integers within a comparatively small range that can easily be manipulated by a computer. It can be helpful, too, if the code values for consecutive letters are consecutive numbers, since this simplifies some operations on text, such as sorting.

Standards

The most important consideration concerning character sets is standardization. Transferring text between different makes of computer, interfacing peripheral devices from different manufacturers, and communicating over networks are everyday activities, and continual translation between different manufacturers' character codes would not be acceptable, so a standard character code is required. The following description of character code standards is necessarily somewhat dry, but an understanding of them is necessary if you are to avoid the pitfalls of incompatibility and the resulting corruption of texts.

Unfortunately, standardization is never a straightforward business, and the situation with respect to character codes remains somewhat unsatisfactory.

ASCII (American Standard Code for Information Interchange) has been the dominant character set since the 1970s. It uses 7 bits

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 207
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

to store each code value, so there is a total of 128 code points. The character repertoire of ASCII only comprises 95 characters, however. The values 0 to 31 and 127 are assigned to *control characters*, such as form-feed, carriage return and delete, which have traditionally been used to control the operation of output devices. The control characters are a legacy from ASCII's origins in early teletype character sets. Many of them no longer have any useful meaning, and are often appropriated by application programs for their own purposes. Table 7.1 shows the ASCII character set. (The character with code value 32 is a space.)

American English is one of the few languages in the world for which ASCII provides an adequate character repertoire. Attempts by the standardization bodies to provide better support for a wider range of languages began when ASCII was adopted as an ISO standard (ISO 646) in 1972. ISO 646 incorporates several national variants on the version of ASCII used in the United States, to accommodate, for example, some accented letters and national currency symbols.

A standard with variants is no real solution to the problem of accommodating different languages. If a file prepared in one country is sent to another and read on a computer set up to use a different national variant of ISO 646, some of the characters will be displayed incorrectly. For example, a hash character (#) typed in the United States would be displayed as a pounds sign (£) in the UK (and vice versa) if the British user's computer used the UK variant of ISO 646. (More likely, the hash would display correctly, but the Briton would be unable to type a pound sign, because it is more convenient to use US ASCII (ISO 646-US) anyway, to prevent such problems.)

➲ The problem of national variants of ISO 646 is particularly acute for programmers. Many programming languages, particularly those whose syntax has been influenced by that of C, make use of some of the characters in the ASCII repertoire which are rarely needed for text as delimiters and operators. For example, the curly brackets { and } are often used to delimit compound statements, function bodies, and other syntactical units; the vertical bar | is used as a component of several operator symbols; the backslash \ is widely used as an escape character within strings. Unfortunately, these otherwise rarely-used symbols are precisely those whose code values correspond to different characters in variants of ISO 646 used outside the US; usually they are used for letters not found in English. Bjarne Stroustrup gives an example of the grief this can cause. The following program:

Table 7.1
The printable ASCII characters

32		33	!
34	"	35	#
36	$	37	%
38	&	39	'
40	(	41	)
42	*	43	+
44	,	45	-
46	.	47	/
48	0	49	1
50	2	51	3
52	4	53	5
54	6	55	7
56	8	57	9
58	:	59	;
60	<	61	=
62	>	63	?
64	@	65	A
66	B	67	C
68	D	69	E
70	F	71	G
72	H	73	I
74	J	75	K
76	L	77	M
78	N	79	O
80	P	81	Q
82	R	83	S
84	T	85	U
86	V	87	W
88	X	89	Y
90	Z	91	[
92	\	93	]
94	^	95	_
96	`	97	a
98	b	99	c
100	d	101	e
102	f	103	g
104	h	105	i
106	j	107	k
108	l	109	m
110	n	111	o
112	p	113	q
114	r	115	s
116	t	117	u
118	v	119	w
120	x	121	y
122	z	123	{
124	|	125	}
126	~		

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 208
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

```
int main(int argc, char* argv[])
{
  if (argc < 1 || *argv[1] == '\0') return 0;
  printf("Hello, %s\n", argv[1]);
}
```

would appear on a terminal using the Danish variant of ISO 646 as[3]

```
int main(int argc, char* argvÆ Å)
æ
  if (argc < 1 øø *argvÆ1Å == 'Ø0') return 0;
  printf("Hello, %sØn", argvÆ1Å);
å
```

A better solution than national variants of the 7-bit ISO 646 character set lies in the provision of a character set with more code points, such that the ASCII character repertoire is mapped to the values 0–127, thus assuring compatibility, and additional symbols required outside the US or for specialized purposes are mapped to other values. Doubling the set of code points was easy: the seven bits of an ASCII character are invariably stored in an 8-bit byte. It was originally envisaged that the remaining bit would be used as a parity bit for error detection. As data transmission became more reliable, and superior error checking was built in to higher level protocols, this parity bit fell into disuse, effectively becoming available as the high order bit of an 8-bit character.

As is customary, the different manufacturers each developed their own incompatible 8-bit extensions to ASCII. These all shared some general features: the lower half (code points 0–127) was identical to ASCII; the upper half (code points 128–255) held accented letters and extra punctuation and mathematical symbols. Since a set of 256 values is insufficient to accommodate all the characters required for every alphabet in use, each 8-bit character code had different variants; for example, one for Western European languages, another for languages written using the Cyrillic script, and so on. [4]

Despite these commonalities, the character repertoires and the code values assigned by the different manufacturers' character sets are different. For example, the character é (e with an acute accent) has code value 142 in the Macintosh Standard Roman character set, whereas it has the code value 233 in the corresponding Windows character set, in which 142 is not assigned as the value for any character; 233 in Macintosh Standard Roman, on the other hand, is È. Because the repertoires of the character sets are different, it is

3 See B. Stroustrup, *The Design and Evolution of C++*, Addison-Wesley, 1994, page 159.

4 Under MS-DOS and Windows, these variants are called *code pages.*

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 209
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

not even always possible to perform a translation between them, so transfer of text between platforms is problematical.

Clearly, standardization of 8-bit character sets was required. During the 1980s the multi-part standard ISO 8859 was produced. This defines a collection of 8-bit character sets, each designed to accommodate the needs of a group of languages (usually geographically related). The first part of the standard, ISO 8859-1 is usually referred to as *ISO Latin1*, and covers most Western European languages. Like all the ISO 8859 character sets, the lower half of ISO Latin1 is identical to ASCII (i.e. ISO 646-US); the code points 128–159 are mostly unused, although a few are used for various diacritical marks; Table 7.2 shows the 96 additional code values provided for accented letters and symbols. (The character with code value 160 is a 'non-breaking' space.)

⇨ The Windows Roman character set is sometimes claimed to be the same as ISO Latin1, but it uses some of the code points between 128 and 159 for characters which are not present in ISO 8859-1's repertoire.

Other parts of ISO 8859 are designed for use with Eastern European languages, including Czech, Slovak and Croatian (ISO 8859-2 or Latin2), for languages that use the Cyrillic alphabet (ISO 8859-5), for modern Greek (ISO 8859-7), Hebrew (ISO 8859-8), and others — there is a total of ten parts to ISO 8859, with more projected, notably an ISO Latin0, which includes the Euro currency symbol.

ISO 8859 has several shortcomings. In practical terms, the worst of these is the continued use of manufacturers' proprietary non-standard character sets. These are firmly entrenched in use and entangled with their corresponding operating systems and other software. Experience shows that, in time, standards are adopted if they can be seen to provide benefits, but there are fundamental problems with 8-bit character sets, which make it seem likely that ISO 8859 will not achieve universal adoption, but that newer standards will render it obsolete. The main problem is simply that 256 is not enough code points — not enough to represent ideographically based alphabets, and not enough to enable us to work with several languages at a time (unless they all happen to use the same variant of ISO 8859).

Table 7.2
The top part of the ISO Latin1 character set

160		161	¡
162	¢	163	£
164	¤	165	¥
166	¦	167	§
168	¨	169	©
170	ª	171	«
172	¬	173	-
174	®	175	¯
176	°	177	±
178	²	179	³
180	´	181	µ
182	¶	183	·
184	¸	185	¹
186	º	187	»
188	¼	189	½
190	¾	191	¿
192	À	193	Á
194	Â	195	Ã
196	Ä	197	Å
198	Æ	199	Ç
200	È	201	É
202	Ê	203	Ë
204	Ì	205	Í
206	Î	207	Ï
208	Ð	209	Ñ
210	Ò	211	Ó
212	Ô	213	Õ
214	Ö	215	×
216	Ø	217	Ù
218	Ú	219	Û
220	Ü	221	Ý
222	Þ	223	ß
224	à	225	á
226	â	227	ã
228	ä	229	å
230	æ	231	ç
232	è	233	é
234	ê	235	ë
236	ì	237	í
238	î	239	ï
240	ð	241	ñ
242	ò	243	ó
244	ô	245	õ
246	ö	247	÷
248	ø	249	ù
250	ú	251	û
252	ü	253	ý
254	þ	255	ÿ

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 210
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Unicode and ISO 10646

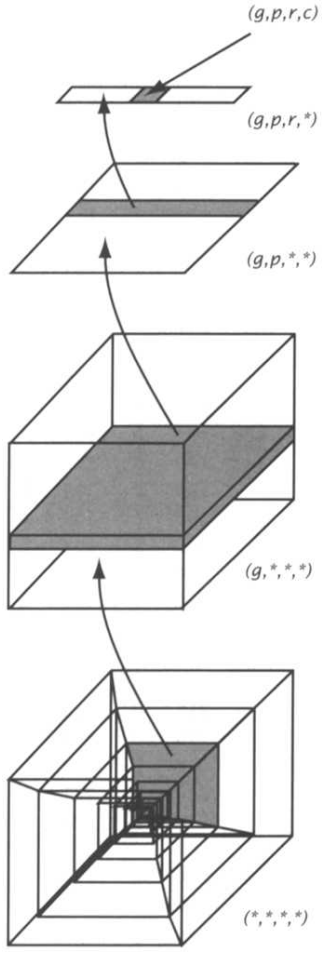


Figure 7.1
The four-dimensional structure of ISO 10646

The only possible solution is to use more than one byte for each code value. A 16-bit character set has 65,536 code points; putting it another way, it can accommodate 256 variants of an 8-bit character set simultaneously. Similarly, a 24-bit character set can accommodate 256 16-bit character sets, and a 32-bit character set can accommodate 256 of those. The ISO (in conjunction with the IEC) set out to develop a 32-bit character set, designated ISO 10646, structured along the lines of the previous two sentences: a collection of 2^{32} characters can be arranged as a hypercube (a 4-dimensional cube) consisting of 256 groups, each of which consists of 256 planes of 256 rows, each comprising 256 characters (which might be the character repertoire of an 8-bit character set). The intention was to structure the immense character repertoire allowed by a 32-bit character set with alphabets distributed among the planes in a linguistically sensible way, so that the resulting character set would have a clear logical structure. Each character can be identified by specifying its group g, its plane p, and a row r and column c (see Figure 7.1). Each of g, p, r and c is an 8-bit quantity, which can fit in one byte; four bytes thus identify a unique character, so, inverting our viewpoint, the code value for any character is the 32-bit value which specifies its position within the hypercube.

To make the structure of the character set evident, we usually write code points as quadruples (g, p, r, c). By extension, such a quadruple also identifies a subset of the character set using a $*$ to denote all values in the range 0–255. Thus $(0, 0, 0, *)$ is the subset with all but the lowest-order byte zero. In ISO 10646 this subset is identical to ISO Latin1.

At the same time as the ISO was developing this elegant framework for its character set, an industry consortium was working on a 16-bit character set, known as *Unicode*. As we noted above, a 16-bit character set has 65,536 code points. This is not sufficient to accommodate all the characters required for Chinese, Japanese and Korean scripts in discrete positions. These three languages and their writing systems share a common ancestry, so there are thousands of identical ideographs in their scripts. The Unicode committee adopted a process they called *CJK consolidation*,[5] whereby characters used in writing Chinese, Japanese and Korean are given the same code value if they look the same, irrespective of

5 Some documents use the name *Han unification* instead.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 211
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

which language they belong to, and whether or not they mean the same thing in the different languages. There is clearly a cultural bias involved here, since the same process is not applied to, for example, upper-case A and the Greek capital alpha, which are identical in appearance but have separate Unicode code values. The pragmatic justification is that, with Chinese, Japanese and Korean, thousands of characters are involved, whereas with the European and Cyrillic languages, there are relatively few. Furthermore, consolidation of those languages would interfere with compatility with existing standards.

Unicode provides code values for all the characters used to write contemporary 'major' languages, as well as the classical forms of some languages. The alphabets available include Latin, Greek, Cyrillic, Armenian, Hebrew, Arabic, Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayalam, Thai, Lao, Georgian, and Tibetan, as well as the Chinese, Japanese and Korean ideograms and the Japanese and Korean phonetic and syllabic scripts. It also includes punctuation marks, technical and mathematical symbols, arrows, and the miscellaneous symbols usually referred to as dingbats (pointing hands, stars, and so on). In addition to the accented letters included in many of the alphabets, separate diacritical marks, such as accents and tildes, are available and a mechanism is provided for building composite characters by combining these marks with other symbols. (This not only provides an alternative way of making accented letters, it also allows for the habit mathematicians have of making up new symbols by decorating old ones.) Code values for nearly 39,000 symbols are provided, leaving some code points unused. Others are reserved for the UTF-16 expansion method (described briefly later on), while a set of 6,400 code points is reserved for private use, allowing organizations and individuals to define codes for their own use. Even though these codes are not part of the Unicode standard, it is guaranteed that they will never be assigned to any character by the standard, so their use will never conflict with any standard character, although it might conflict with other individuals'.

Unicode is restricted to characters used in text. It specifically does not attempt to provide symbols for music notation or other symbolic writing systems that do not represent language.

Unicode and ISO 10646 were brought into line in 1991 when the ISO agreed that the plane $(0, 0, *, *)$, known as the *Basic Multilingual Plane (BMP)*, should be identical to Unicode. ISO 10646 thus utilizes

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 212
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

CJK consolidation, even though its 32-bit code space does not require it. The overwhelming advantage of this arrangement is that the two standards are compatible (and the respective committees have pledged that they will remain so). To understand how it is possible to take advantage of this compatibility, we must introduce the concept of a character set *encoding*.

An encoding is another layer of mapping, which transforms a code value into a sequence of bytes for storage and transmission. When each code value occupies exactly one byte, it might seem that the only sensible encoding is an identity mapping: each code value is stored or sent as itself in a single byte. Even in this case, though, a more complex encoding may be required. Because 7-bit ASCII was the dominant character code for such a long time, there are network protocols that assume that all character data is ASCII, and remove or mangle the top bit of any 8-bit byte. To get round this limitation, it may be necessary to encode 8-bit characters as sequences of 7-bit characters. One encoding used for this purpose is called *Quoted-Printable (QP)*. This works quite simply: any character with a code in the range 128–255 is encoded as a sequence of three bytes: the first is always the ASCII code for =, the remaining two are the codes for the hexadecimal digits of the code value. For example, é, has value 233 in ISO Latin1, which is E9 in hexadecimal, so it is encoded in QP as the ASCII string =E9. Most characters with codes less than 128 are left alone. An important exception is = itself, which has to be encoded, otherwise it would appear to be the first byte of the encoded version of some other character. Hence, = appears as =3D.

To interpret a sequence of bytes correctly we need to know which encoding and which character set is being employed. Many systems rely on convention, or stipulate (or quietly assume) that all text is encoded in ISO Latin1. A more flexible approach is to use an additional feature of the MIME content type specifications which we introduced in Chapter 2. Following the type and subtype, a character set specification of the form ; `charset` = *character set* may appear. A set of 'official' character set names is maintained by IANA. For example, the type of a Web page composed in ISO Latin1 is

```
text/html; charset = ISO-8859-1
```

This information should be included in HTTP response headers and MIME-encoded email messages. It can also be specified using appropriate tags in a mark-up language, as explained in Chapter 8.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 213
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

For ISO 10646 the obvious encoding scheme, known as UCS-4, employs four bytes to hold each code value. Any value on the BMP will have the top two bytes set to zero; since most values that are currently defined are on the BMP, and since economic reality suggests that for the foreseeable future most characters used in computer systems and transmitted over networks will be on the BMP, the UCS-4 encoding wastes space. ISO 10646 therefore supports an alternative encoding, UCS-2, which does the obvious thing and drops the top two bytes. UCS-2 is identical to Unicode.

Unicode encodings go further. There are three *UCS Transformation Formats (UTFs)* which can be applied to Unicode code values. UTF-8 takes the reasoning we just applied to 32-bit values a step further. ASCII code values are likely to be more common in most text than any other values. Accordingly, UTF-8 encodes UCS-2 values so that if their high-order byte is zero and the low-order byte is less than 128, the value is encoded as the single low-order byte. That is, ASCII characters are sent as themselves. Otherwise, the two bytes of the UCS-2 value are encoded using up to six bytes, with the highest bit of each byte set to 1 to indicate it is part of an encoded string and not an ASCII character. Text encoded with UTF-8 is thus a string of 8-bit bytes, and is therefore vulnerable to mangling by protocols that can only handle ASCII. UTF-7 is an alternative encoding which uses a technique similar to that described for QP to turn Unicode characters into streams of pure ASCII text, which can be transmitted safely.

The UTF-16 encoding has a different emphasis. This encoding allows pairs of 16-bit values to be combined into a single 32-bit value, thus extending the repertoire of Unicode beyond the BMP. Only values in a limited range can be combined this way, with the result that UTF-16 only provides access to an additional 15 planes of the full ISO 10646 character set. These comprise nearly a million characters under UTF-16, which seems to be sufficient for present purposes.

To summarize: ISO 10646 is a 32-bit character code, arranged in 256 groups, each of which consists of 256 planes accommodating 65,536 characters each. The UCS-4 encoding utilizes four bytes to hold the full 32-bit code value for any character; the UCS-2 encoding utilizes just two bytes, to hold 16-bit values for characters on the $(0, 0, *, *)$ plane. UCS-2 is identical to Unicode. ISO Latin1 is the eight bit code equivalent to the $(0, 0, 0, *)$ row of ISO 10646, and ASCII is a subset of ISO Latin1. UTF-8 allows any ISO 10646 or

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 214
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Unicode value to be encoded as a sequence of 8-bit bytes, such that ASCII values are left unchanged in a single byte. UTF-16 is an extension mechanism which provides Unicode with access to an extra 15 planes of the full ISO 10646 character set.

The markup languages HTML and XML, described in Chapters 8 and 13, and the Java programming language use Unicode as their character set. Once it also becomes adopted as the native character set of the major operating systems, the task of transferring text that uses characters beyond the US-Anglophone ASCII repertoire should be substantially eased.

Fonts

To display a piece of text, each stored character value must be mapped to a visual representation of the character's shape. Such a representation is called a *glyph*. As Figure 7.2 shows, a single character can be represented by a multitude of different glyphs. Many small details can be changed without destroying the fundamental identity of the abstract character being represented. We can recognize all the glyphs in Figure 7.2 as lower-case letter q. In addition to the variations in shape which that figure illustrates, glyphs can vary in size, from tiny subscripts to banner headlines over an inch tall. If we are to be able to use glyphs systematically, it is necessary to impose some organization on them.

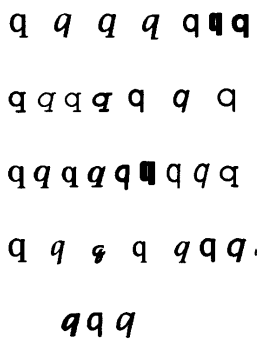


Figure 7.2
Glyphs for lower case q

Glyphs are arranged into collections called *fonts*. The name has been taken from traditional printing. In the letterpress (or 'cold metal') technique a page is assembled out of individual pieces of type, each one of which is a metal block, with a mirror image of a letter or other symbol in relief on one end. These blocks are clamped in a frame together with spacers, and pages are printed by inking the faces of the type and pressing the paper against them. In this context, a font is a collection of pieces of type — actual metal objects. Usually, the typesetter works from a cabinet in which the type is arranged in drawers holding multiple instances of each letter shape at each size that is required. All the shapes in a particular font will have been made from masters produced by a type designer so that they share certain visual characteristics, and combine well with each other to make a harmonious and readable page.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 215
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Letterpress has been largely superseded, first by mechanical hot metal technologies, such as Monotype and Linotype machines, and now by digital typesetting. The concept of a font has been retained through these technological changes, although its realization has evolved, first into a set of moulds into which hot lead is poured to make letters for typesetting machines, and then into a computer file that holds a description of the graphic forms of the glyphs. The idea that a font combines a set of glyphs that are visually related and designed to work together has not changed. In fact, many of the fonts available for use on computers today are versions of traditional typesetters' fonts, some of which are based on original designs from the 15th century.

Accessing Fonts

Text originates as a stream of character codes. To turn it into a form that can be displayed, these must be replaced by glyphs chosen from one or more fonts. Once this replacement has been performed, the content of the text (the character codes) becomes lost and can only be retrieved with difficulty. Essentially, the text turns into graphics, which cannot easily be searched or edited. (A person can still read the text, of course — in fact, they can't read it in any other form — but a program cannot manipulate it in any way.) What is more, since a stream of character codes is much more compact than an image representing the same text, replacement of character codes by glyphs also increases the size of the text. The graphical representation of text is inefficient because it takes no advantage of the repetitive nature of text: the same character will typically occur many times, so if glyphs are substituted, many copies of the same glyph will be used, and each will be much larger than the corresponding character code.

⇨ There is an exception to our observation that text originates as a stream of character codes. If a page of printed text is scanned the result is a bitmapped image of the page. If you want to get hold of the characters from a scanned text you need to use special *optical character recognition (OCR)* software, which uses pattern recognition techniques to analyse the shapes within the bitmapped image and deduce the characters they represent. OCR is a mature technology, but 100% accurate recognition is rarely achieved.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 216
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Because of these considerations, it is normal to keep text in some character-based form, and only to use glyphs when it is actually displayed or incorporated into a graphic image. The simplest case is that of *monostyled* text: text that is displayed in a single font. In that case, a text file need only contain the characters of the text itself: it is the familiar 'plain ASCII' (or perhaps plain ISO 8859) file. When such a file is displayed, the character codes are used to select glyphs from a font, either a default system font, or a single font selected by the user from within the program being used to display the text.

Monostyled text is rarely adequate for multimedia production, where a richer typographic experience is usually required. The principle of selecting glyphs using character codes remains the same, but additional information is needed to control the selection of fonts. We will describe various forms this information may take in Chapter 8. At this point, we must consider the question: Where might fonts be found?

There are only two possibilities. Glyphs must be taken either from fonts stored on the system[6] being used to display the text, or from fonts embedded in the text file. In the latter case, the fonts must originally have been stored on the system used to prepare the text, and therein lies the most important advantage of this approach: multimedia designers can use fonts of their own choosing, confident that they will be available when they are needed, because they are embedded in the same file as the text. If they are not, then only the fonts on the user's system will be available, and there is no reason to suppose that the user's selection will be the same as the designer's, so there is a possibility that some font required by the text will not be available. The probability of this occurring increases if the designer chooses esoteric fonts — as designers sometimes will.

Why then might you prefer not to embed fonts in your text files? One potent reason is that not all text file formats allow you to do so. In particular, HTML presently has no facility for embedding fonts,[7] although, in conjunction with CSS stylesheets, it does allow you to specify the fonts you wish to be used to display textual elements of Web pages. The other reason for preferring to rely on fonts on the user's system is that fonts are fairly bulky objects: the fonts used for this book, for example, vary in size between about 32 and 50 kilobytes each. It is quite easy to find yourself using half a dozen fonts on a page, so including the fonts with the text can lead to bloated files and consequently to extended download times.

6 Exactly what is meant by 'stored on the system' is system-dependent: on a MacOS system, for example, fonts must be placed in the `Fonts` sub-folder of the `System Folder` (unless a font management utility is being used); on a system running X11 under Unix, fonts must be in some directory included on the user's FONTPATH.

7 Although several mechanisms for downloading fonts have been proposed, none has been adopted as a standard at the time of writing.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 217
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

If a font is not available, various more or less undesirable consequences can result. In the extreme, the result might be a system crash. More likely, text in the missing font will not be displayed, or it will be displayed in some other font that has been substituted. The last case is the least serious, and the most common, but it is far from ideal. As we will see in the next section, every font has its own distinctive character, so substituting another in its place can seriously impair a carefully thought out design. Furthermore, the widths of the glyphs in the substituted font will probably be different from those in the intended font, so that the positioning of individual glyphs on the page will be incorrect. Depending on how the layout of text for display is done, this might result in ragged margins where straight ones were intended, uneven gaps between words, or gaps within words. Any of these defects will have a detrimental effect on the appearance of your text, and may make it hard to read.

Classification and Choice of Fonts

There are thousands of fonts available, each with its own special personality. Some broad characteristics are used to help classify them.

A major distinction is that between *monospaced* (or *fixed-width*) and *proportional* fonts. In a monospaced font, every character occupies the same amount of space horizontally, regardless of its shape. This means that some letters have more white space around them than others. For example, the narrow shape of a lower-case l must be surrounded with white to make it the same width as a lower-case m. In contrast, in a proportional font, the space each letter occupies depends on the width of the letter shape. Paradoxically, this produces a more even appearance, which is generally felt to be easier to read in most contexts. In addition, it allows you to fit more words on to a line. Text in a monospaced font looks as if it was produced on a typewriter or a teletype machine. It can sometimes be used effectively for headings, but is especially suitable for typesetting computer program listings. It is also useful for conveying a 'low-tech' appearance, for example in correspondence where you wish to convey a slighty informal impression. Text in a proportional font has the appearance of a traditional book, and is usually preferred for setting lengthy texts. It is generally felt to be more readable, since letters appear to be

```
Monospaced Font:  Lucida
Typewriter
Each letter occupies the
same amount of
horizontal space, so
that the text looks as
if it was typed on a
typewriter.
```

Figure 7.3
A monospaced font

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 218
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

tightly bound together into words. Figures 7.3 and 7.4 illustrate the difference between monospaced and proportional fonts.

Proportional Font: Lucida Bright
Each letter occupies an amount of horizontal space proportional to the width of the glyph, so that the text looks as if it was printed in a book.

Figure 7.4
A proportional font

Probably the most widely used monospaced font is Courier, which was originally designed for IBM typewriters, and has achieved wide currency because it is one of the fonts shipped as standard with all PostScript printers. Many proportional fonts are in common use. As you can no doubt discern, the font used for the main text in this book, Lucida Bright, is proportionally spaced. Most of the classical book fonts, such as Times, Baskerville, Bembo, and Garamond are also proportional, as are many newer fonts, such as Helvetica.

Another very broad distinction is between *serifed* and *sans serif* (sometimes *sanserif*) fonts. *Serifs* are the little strokes added to the ends of character shapes (see Figure 7.5). These strokes are present in serifed fonts, but omitted in sans serif fonts, which consequently have a plainer look. Serifs originate in marks produced by chisels on Roman stone inscriptions, so serifed fonts are sometimes called Roman fonts. Figure 7.6 shows some text in Lucida Sans, a sans serif version of the Lucida Bright font.

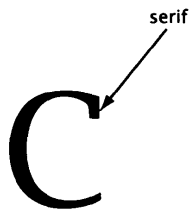


Figure 7.5
Serifs

Sans Serif Font: Lucida Sans
The letters of a sans serif (or sanserif) font lack the tiny strokes known as serifs, hence the name. They have a plain, perhaps utilitarian, appearance.

Figure 7.6
A sans serif font

Sans serif fonts are a comparatively new development, on a typographical time scale, and have only gradually gained acceptance for general use — in some typographical catalogues they are still identified by the alternative name of *grotesques*. In the nineteenth century, sans serif fonts were indeed rather grotesque, being crude designs used mostly for advertising and posters. Only in the twentieth century have they become accepted for use in books, with the development of more elegant and refined designs. The best known sans serif font is probably Helvetica, which again is one of the standard PostScript fonts. Other sans serif fonts you may meet include Univers and Arial (the latter popularized by Microsoft). Gill Sans, another very popular sans serif font, was based on the font used for the signs on the London Underground, which had helped to generate interest in sans serif type.

There is contradictory evidence as to whether serifed or sans serif fonts are more readable — spacing probably makes more difference. However, serifs are very small features, and therefore difficult to render accurately at low resolutions, which can mean that text in a serifed font is hard to read on a computer screen, simply because the letters are not being accurately reproduced. Sans serif fonts are widely used for such features as window titles and menu entries on these grounds.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 219
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Spacing and serifs are independent properties: sans serif and serifed fonts can equally well be either monospaced or proportional.

A third classification of fonts is based on broad categories of *shape*. In particular, we distinguish between fonts with an *upright* shape, and those with an *italic* shape. Upright fonts, as the name implies, have characters whose vertical strokes (stems) are indeed vertical. Italic fonts imitate a certain style of handwriting, and have letters that are slanted to the right. Additionally, the letter shapes in an italic font are formed differently from those in an upright font, so that they share some of the characteristics of italic handwriting. When digital fonts first appeared, 'italic' fonts were sometimes produced simply by applying a shear transformation to an upright font. The effect is rather different, since the calligraphic features of true italic fonts are missing. Such fonts are now used in their own right, not as substitutes for italics. They are said to have a *slanted* shape. The difference is illustrated in Figures 7.7 and 7.8, which show italic and slanted versions of the upright font used for the main text of this book.

Italic Font: Lucida Bright Italic
The letters of an italic font slope to the right, and are formed as if they were made with an italic pen nib. Italics are conventionally used for emphasis, and for identifying foreign words and expressions.

Figure 7.7
An italic font

Most italic fonts are variations on or designed as companions to upright fonts. For example, Times Italic is an italic version of Times Roman. There are, however, some fonts with an italic shape which are designed on their own; these are usually intended to have the character of handwriting, and to be used where something more human than a conventional typeface is desired. Among the best known calligraphic fonts are several versions of Chancery, including Zapf Chancery. Specialist fonts based on samples of real handwriting are also available. Figures 7.9 and 7.10 are examples of a relatively formal calligraphic font and a handwriting font.

Slanted Font: Lucida Bright Oblique
The letters of a slanted font share the rightward slope of italic fonts, but lack their calligraphic quality. Slanted fonts are sometimes used when a suitable italic font is not available, but may be preferred to italics when a more modern look is wanted.

Figure 7.8
A slanted font

⇨ One type of font that doesn't quite fit our classifications is the *small caps* font, in which small versions of the upper case letters are used as lower case letters. This variation is sometimes considered to be a shape, although the sense is somewhat different from that in which italic is considered a shape. Small caps fonts are most useful for trademarks and so on, which are normally written entirely in upper case: full size upper case words stand out awkwardly in ordinary text, but small caps are less intrusive.

Calligraphic Font: Lucida Calligraphy
Calligraphic fonts usually resemble 'round hand' or 'copperplate' handwriting, unlike italic fonts.

Figure 7.9
A calligraphic font

Digital technology has made some shapes widely available that previously could not be easily produced. In particular, outline fonts, which have hollow letters, and fonts with drop shadows, as illustrated in Figure 7.11, can be easily constructed using graphics

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 220
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Handwriting Font: Lucida Handwriting
Handwriting fonts are based on samples of real people's handwriting, so they are often quite idiosyncratic.

Figure 7.10
A handwriting font

Drop shadows create a slight 3-D effect

Figure 7.11
A font with drop shadows

effects. Such fonts should be viewed as gimmicks and used judiciously, if at all.

Some fonts appear somewhat squashed horizontally, compared with the normal proportions of most fonts. They are referred to as *condensed* fonts, and are intended for applications, such as marginal notes or narrow newspaper columns, where it is desirable to be able to fit text in as tightly as possible. In contrast, some fonts, described as *extended* are stretched out horizontally, making them more suitable for headings, and other isolated text elements.

Finally, fonts can be classified according to their *weight*, that is, the thickness of the strokes making up the letters. Thicker strokes make text look darker and more solid. Conventionally, we call fonts with a heavy weight (thick strokes) *boldface* or simply *bold*. Like italics, bold fonts are usually versions of other fonts with a lighter weight. As a result, boldness is not an absolute property: a bold version of a font whose normal weight is particularly light may be lighter than the normal version of a heavy font, whose own bold version will be even heavier. Some fonts may exist in several versions, exhibiting varying degrees of boldness. Under these circumstances, individual styles are described by terms such as *ultra-bold*, *semi-bold*, *light* and *ultra-light*.

Conventional typesetting wisdom has it that boldface is intrusive, and should be reserved for headings and similar uses. Because of the limitations of computer displays, it is sometimes advisable to use boldface more widely than would be appropriate in text intended for paper. For example, conventionally you *never* use boldface for emphasis, always italics. However, italic text, because of its slant, often renders badly at low resolutions, making it hard to read. Under these circumstances, the use of bold text for emphasis may be justified. Bold fonts are also used quite widely for window titles and menu items, because they show up well.

⇨ Word processors often treat underlining as a styling option similar to italicization and emboldening, so you might think that there would be underlined versions of fonts, too. Underlined fonts are extremely rare, though: underlining is scorned in typographic circles, where it is considered to be a poor substitute for italicization, only suitable for typewriters which lack italics. Like most conventional typographical wisdom, the undesirability of underlining is no longer unquestioned. Since more flexible effects can be produced by combining ordinary fonts with lines of various thicknesses, this questioning has not led to an outbreak of underlined fonts.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 221
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Because it makes sense to talk about an italic version or a bold version of an upright font, fonts can be grouped into families.[8] Usually, an upright serifed font is considered to be the normal form, which is augmented by versions in different weights, perhaps an italic form or a sans serif form. Variations are often combined to produce additional versions, such as bold italics, or slanted sans serif. For example, the Lucida Bright family consists of twenty fonts:[9] Lucida Bright is the upright serifed font you are reading now; it is also available in bold, italic, and bold italic. Lucida Sans is the sans serif version, which comes in the same four versions, as do Lucida Typewriter, a fixed width Lucida font, and Lucida Fax, a variant form designed to be especially readable at low resolution. Three calligraphic fonts and a slanted font complete the family. All twenty fonts share a similar feel so that they can be combined without any obtrusive visual discontinuities. In contrast, when fonts from different families are combined, their differences can be very noticeable, as Figure 7.12 shows. Traditionally, such discontinuities have been carefuly avoided, but in recent years designers have taken advantage of the ease with which desktop publishing software allows them to combine fonts to explore new combinations that defy the established conventions.

8
A font family corresponds closely to what is traditionally called a *typeface*. A font is a particular style of some typeface. In pre-digital typography each different size would be considered a separate font, but nowadays we usually consider that the same font can be rendered at different sizes.

9
Actually, there are 25, the additional ones are for mathematics.

Lucida Bright goes well with *Lucida Bright italic,* ***and bold italic,*** but not nearly so well with Palatino.

Figure 7.12
Mixing fonts from different families

In the days of letterpress and hot metal typesetting, to produce text at different sizes required different sets of type, which would qualify as separate fonts. Only a limited number of different sizes could easily be made available. With digital fonts, arbitrary scaling can be applied to glyphs, so that fonts can be printed at any size. Purists maintain that simple scaling is not adequate, and that letter shapes should be designed to look their best at one particular size, and should only be used at that size. Contemporary font technology, however, tends to the idea that a font should exist at only one size and should be scaled to produce any other size that is needed. Special information contained in the font is used to help maintain a pleasing appearance at all sizes, as we will describe in a later section.

As well as the objective factors of spacing, serifs, shape, weight and size that can be used to classify type, there is a more subjective classification based on the sort of jobs for which a font is most suitable. The basic distinction is between *text fonts* and *display fonts*. The terminology is slightly silly, since all fonts are used for text. The distinction is between fonts suitable for continuous text, such as the body of a book or article, and those suitable for

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 222
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

short pieces of isolated text, such as headings, signs or advertising slogans on posters.

⇨ Sometimes, a finer distinction is drawn within the class of display fonts. *Decorative* fonts are those for which appearance is the primary design consideration. They often incorporate ornaments and other features that make them quite unsuitable for extended use. *Headline* fonts, as the name implies, are designed for use in headlines and other situations where it is important to attract the reader's attention. This leaves a category to which the name display fonts is often attached, consisting of fonts intended especially for use at large sizes, with other features designed to take advantage of the possibilities offered by large characters; for example the serifs might be especially fine.

Text fonts must be unobtrusive, so that they do not intrude on the reader and interfere with the primary message of the text. They must also be easy to read, so that they do not cause fatigue when they are read for hours at a time. To some extent, whether or not a font is intrusive depends on whether it is familiar; at the same time, the criteria for selecting text fonts have not changed over the years. The combination of these factors means that text fonts tend to be conservative. Invariably they are upright, more often serifed than not, and of a medium weight.

A Display Font: Bodoni Highlight

Display fonts are designed for short pieces of text, such as headlines. They are not intended for use in lengthy passages.

Figure 7.13
A display font

Display fonts are another matter. Here, the intention is to get across a short message. Garish design that would be offensive in a text font becomes eye-catching; innovation attracts attention. There is a far greater diversity of design among display fonts than among text fonts (Figure 7.13 shows just one example), and whereas the same text fonts continue to be used year after year, display fonts are subject to fashion.

⇨ Digital font technology has made the mechanics of producing new fonts much simpler than it was when type had to be created out of metal. It has also made possible some effects that could not previously be physically achieved. Consequently, recent years have seen an acceleration in the rate of innovation in font design, and the introduction of aggressively unconventional display fonts. Opinion is predictably divided over the virtues of 'grunge typography', as this new design movement is sometimes called.[10] In some circles it is condemned as unreadable and ugly, while in others it is considered to be the most exciting development in type design since the invention of the printing press. The ease with which grunge typefaces have been absorbed into mainstream advertising suggests that they might not be especially revolutionary after all.

10
Although you probably won't be surprised to learn that it is also sometimes called 'post-modern' or (what else?) 'deconstructivist'.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 223
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Conventional ideas about font usage are based on centuries of experience of making books, pamphlets, posters, packaging, road signs, shop fronts and other familiar forms of printed matter. While much of this experience can be applied to the textual components of multimedia, some aspects of the new media demand a fresh approach. Generally, display fonts can be used on a computer monitor in much the same way as they can on paper, so where text is combined with images, fonts can be used as they would be on a poster; attention-grabbing headings can effectively use the same display fonts as you might find in a book. Continuous text is more problematical: the low resolution of most monitors can lead to distortion of letter shapes, making fonts that work well on paper hard to read, especially at small sizes. The obvious solution is to use fonts at larger sizes than is customary in books, and this is often done: text for electronic display is often set as much as 60% larger than text in ordinary books. An alternative is to look for a font that has been designed to be readable at low resolution; Monaco and Arial are examples of such fonts. Sans serif fonts tend to survive better at low resolutions, which makes them more suitable for this purpose.

The way the text in multimedia productions is arranged gives rise to some less technical questions concerning fonts. Long continuous passages covering many pages are cumbersome and tiring to read on a screen, and do not integrate well with other media. It is common, therefore, to find multimedia text elements constructed as small pieces that fit onto one or two screen-fulls. These pieces will often resemble, in form and style, the short explanatory placards attached to museum displays and zoo exhibits. This sort of text partakes of the character of both continuous text for which we would use a text font — it has a non-trivial extent and content — and the shorter texts normally set in a display font — it is succinct and usually has a message to get across. A restrained display font will often work well in this situation, as will some quite unconventional solutions, such as a text font in a large bold version, or coloured text on a contrasting background.

Text for multimedia is often prepared using the same tools as are used in 'desktop publishing' (DTP). It is important to remember that the output from conventional DTP programs is intended for printing on paper. We reiterate that text that will look excellent when printed on paper may well be unreadable when viewed on a computer monitor. It is necessary to adapt the way you use these

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 224
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

tools, bearing in mind the actual medium for which your output is destined, or to use tools that have support for output to alternative media.

One final consideration which is unique to digital text is that, in many cases, the multimedia designer has no control over the fonts that will be used when text is finally displayed. As we explained previously, it may be necessary for completely different fonts to be substituted on different computer systems; in some cases, the software used for display may let users override the original fonts with those of their own choosing. Consequently, unless you know that neither of these circumstances will arise, there is no point in carefully exploiting the features of a particular font to achieve some special effect — the effect may never be seen by anyone except you.

Most of the preceding description of fonts has concentrated on letters, implicitly in the Latin alphabet. However, as the discussion of character sets in the previous section indicated, we use far more characters than these letters, and just as we need character sets to provide code values for a large character repertoire, so we need fonts to provide glyphs for them. In fact, you can think of a font as a mapping from abstract characters to glyphs, in much the same way as a character set is a mapping from abstract characters to code points. Like a character set, a font is only defined for a specific character repertoire. Most fonts' repertoires consist of the letters from some alphabet, together with additional symbols, such as punctuation marks. You might be forgiven for hoping that fonts would be structured around one of the character set standards, such as ISO Latin1, but this is not usually the case. It is generally possible to access the individual glyphs in a font using a numerical index, but, although the printable ASCII characters are usually in their expected positions, other symbols may well be in positions quite different from that corresponding to their code values in any ISO character set.

The way in which glyphs are grouped into fonts owes more to printing tradition than to the influence of character code standards, with the alphabet being the focus of design. Even then, some fonts may not include lower case letters, if the designer intended the font only to be used for headlines, for example. Specialized non-alphabetic symbols, such as mathematical symbols are usually grouped into their own fonts, known as *symbol fonts* or *pi fonts*. Some fonts consist entirely of graphic images: you may encounter fonts of arrows, or fonts of pointing hands, for example. Some

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 225
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

images are considered to be symbols and are included in character sets — fonts containing these symbols are called *dingbat fonts* — but others are put together by font designers for special applications, and do not fit into any character set framework.

⇨ A particularly interesting example is available as we write (in mid-1998): the impending adoption of a single currency, the Euro, by most of the member states of the European Union has led to the design of a new currency symbol to designate it. This symbol (which looks a bit like €) is not present in any fonts predating the Euro. Some font vendors have responded by producing fonts consisting of nothing but Euro symbols, in different styles. Instead of choosing the Euro symbol from your text font, you use the Euro font and choose whichever version most closely matches the style of the surrounding font.

As a result of all this, another level of encoding is required in order to map stored character codes in some character set to glyphs. Additionally, some mechanism may be needed to combine separate fonts, for example, an alphabetic font and a symbol font, to provide glyphs for all the characters used in a document — a problem which will become acute as the wider range of characters provided by Unicode comes into wider use. Fortunately, the mechanisms for accessing glyphs are usually handled transparently by text layout software.

Font Terminology

Typography has its own specialized vocabulary, some of which it is necessary to understand if you are to find your way around among fonts and their descriptions.

Much of the description of a font's characteristics consists of measurements. These are usually given in units of *points (pt)*. In digital typography, one point is 1/72 of an inch, which makes 1pt equal to just under 0.3528mm. A point is thus a very small unit, suitable for measuring the dimensions of such small objects as typeset characters. For slightly larger quantities, such as the distance between lines of text, we often use a *pica (pc)*, which is equal to 12pt (or 1/6in or 4.2333mm).

⇨ Unlike other units of measurement, such as the metre or the foot, the point does not have an internationally agreed standard magnitude. The value of exactly 1/72in is a comparatively recent

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 226
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

innovation, pioneered by PostScript. In English-speaking countries, the printer's point is 1/72.27in, and some software, such as TEX, still uses this value by default. In France, the Didot point, which is about 7% larger than the printer's point, is preferred.

A font's size is quoted in points, as in '12pt Times Roman' or '10pt Lucida Sans'. The value specified is technically the font's *body size*. In the days of letter press, the body size was the height of the individual pieces of metal type, each of which were the same size so that they could be fitted together. Hence, the body size was the smallest height that could accommodate every symbol in the font. This remains the case: the size is not necessarily equal to the height of any character in the font (although parentheses are often as tall as the body size), it is the height between the top of the highest character and the bottom of the lowest.

⇨ This is often the case, but the body height is actually an arbitrary distance, chosen to provide a suitable vertical space for the letters to sit in when they are set in lines. A font with a particularly open character might have a body size somewhat larger than is necessary just to accommodate all its characters, so that there is always some extra space around them when text is set in that font on conventional baselines.

In normal text, characters are arranged so that they all sit on the same horizontal line, just as, when we write on lined paper, we ensure that all our letters are on the lines. This line is called the *baseline*. An important font dimension is the height between the baseline and the top of a lower-case letter x. This value is the font's *x-height*; the bodies of most lower-case letters fit in between the baseline and the x-height. Some letters, such as h, have strokes that rise above the x-height; these are called *ascenders*. Similarly, letters such as y extend below the baseline; the extending strokes are called *descenders*. These terms are illustrated in Figure 7.14. Sometimes, the size of the largest ascender (the distance between the x-height and the top of the body) is called the *ascent*, and the corresponding descending dimension is called the *descent* of the font.

Ascenders are not the only things that extend above the x-height: capital letters do so as well. In many fonts, though, the capitals do not extend to the full height of ascenders, so we need another quantity, the *cap height*, also shown in Figure 7.14, to characterize the vertical extent of capitals. (In English text, the difference between cap height and ascent is most obvious when the combination 'Th' occurs at the beginning of a sentence.)

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 227
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

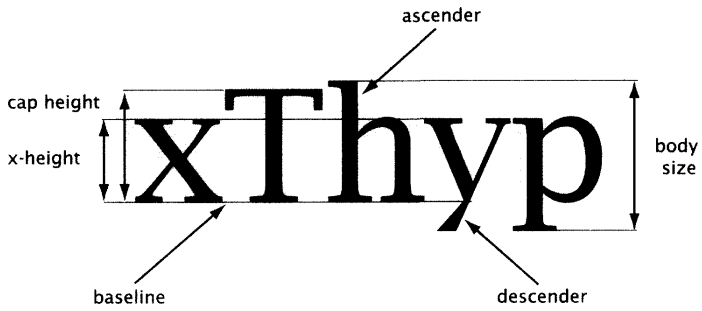


Figure 7.14
Some font terminology

The ratio of the x-height to the body size is one of the most important visual characteristics of a font. Since the bulk of the text lies between the baseline and the x-height, a font with a relatively high x-height will look bigger than a font with a lower x-height at the same size. If you look back at Figure 7.12, you will be able to see that one of the things that prevents the text in Garamond merging happily with that in Lucida Bright is that the latter font has a much higher x-height. This feature is common to the entire Lucida family, so the Lucida Italic mixes easily with the upright Lucida Bright.

⇨ If you look closely at Figure 7.14 you will see that the curved tops of the h and p actually extend through the x-height. This phenomenon is called *overshoot*, and helps make the letters look more uniform in size. The extent of the overshoot is another factor that characterizes the look of a font.

The x-height of a font is used as a unit of measurement, usually written ex. It has the useful property of not being an absolute unit, like a point, but a relative one; it changes with the font's size, and is different for different fonts, but always stands in the same relation to the height of lower-case letters, so it provides a convenient way of expressing vertical measurements that should change in proportion to this quantity. A similar unit of horizontal distance is the em. Traditionally, 1em is the width of a capital letter M. In many fonts, M is as wide as the body size, so the meaning of 1em has migrated over the years, and is now often taken as a unit of length equal to the font size;[11] for a 10pt font, 1em is equal to 10pt, for a 12pt font it is equal to 12pt, and so on. Long dashes — like these — used for parenthetic phrases are 1em long, so they are called *em-dashes*. You will sometimes see another relative unit, the *en*, which is the width of a capital N, and usually defined as 0.5em. An *en-dash* is 1en long; en-dashes are used for page or date ranges, such as 1998–99.[12]

11 The em unit is defined like this in CSS (see Chapter 8), for example.

12 There are no fixed rules about dash usage. Some publishers stipulate the use of en-dashes for all purposes. You should never use a hyphen instead of a dash, though.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 228
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Other features which characterize the look of a font include the size and shape of serifs, and the ratio of the thickness of thick and thin strokes. For example, so-called *modern* fonts (actually based on designs about 200 years old) are characterized by high contrast between the thick and thin strokes, and serifs without brackets (the curves that join the serif to its stem). Fans of these fonts[13] believe these features produce a brilliant, sophisticated appearance on the page, while their detractors find them affected and illegible, compared to the *old-style* fonts with their more uniform solid appearance, or the cleaner lines of twentieth century sans serif fonts. The implications for display at low resolution should by now be clear.

13
Yes, fonts have their fans. Indeed, once you start to look closely at fonts, you may find you start to develop very strong feelings about them. You have been warned.

The features of individual letters are not all we need to consider in a font. We also have to look at how letters are combined. As we stated earlier, most text fonts are proportionally spaced, with each letter occupying as much horizontal space as it needs to. As Figure 7.15 shows, each letter has a *bounding box*, which is the smallest box that can enclose it. Normally, a slight gap is left between the bounding boxes of adjacent characters. In other words, when drawing a glyph immediately following another one, it is drawn relative to a *glyph origin*, which lies outside of the bounding box, usually to its left, by a distance known as the character's *left side bearing*, as shown in Figure 7.15 (where we have considerably exaggerated the magnitude of the side bearing).

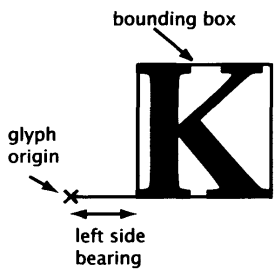


Figure 7.15
Side bearings

Sometimes, when two particular letters are placed next to each other, the total amount of space between them looks too great or too small. Typographers normally adjust the spacing to make it look more uniform. (Once again, you will notice the apparent contradiction that it is necessary to introduce non-uniformity to achieve the subjective appearance of uniformity.) This process of adjustment is called *kerning*, and is illustrated in Figure 7.16. (Look closely, kerning is subtle.) The kerning pairs for a font are defined by its designer. Carefully produced fonts may have hundreds of kerning pairs, each with its own spacing. The information about which pairs of characters are to be kerned, and by how much, is stored as part of the font.

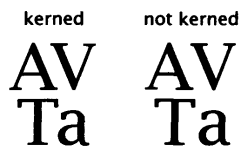


Figure 7.16
Kerning

The measurements which describe the size of individual characters and the spacing between them are collectively known as *font metrics*. Programs that perform typesetting need access to font metric information, in order to determine where to place each glyph as they build up lines, paragraphs and pages. The organization of

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 229
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

this information depends on the font format, computer system and typesetting software being used.

⇨ Two operations related to kerning are *letter spacing* and *tracking*. Letter spacing is the process of changing the space between all letters in a font, stretching or squeezing the text. This may be done either to change the appearance of the font, or to fit an awkward line of text on to a page. (The latter operation is severely frowned on in the best typesetting circles.) Tracking is a systematic application of letter spacing depending on the font size: large letters need proportionally less space between them to look right.

Certain character combinations just will not look right, no matter how you space them. Printers traditionally replace occurrences of such troublesome sequences by single composite characters, known as *ligatures*. In English text, commonly used ligatures include ff, fl, fi and ffl. There is no standard set of ligatures, and some fonts provide more of them than others. Monospaced fonts do not usually provide any. Figure 7.17 shows why we need ligatures, and the improvement achieved by using them. Ligatures are stored as extra characters in the font.

without: fine fluffy soufflés
with: fine fluffy soufflés

Figure 7.17
Ligatures

High quality text layout software will automatically deal with kerning and ligatures, making the appropriate spacing corrections and character subsitutions when it encounters the codes for characters that require such treatment. Word processors and Web browsers cannot generally do this.

Digital Font Technology

Glyphs are just small images, which can be stored in a font using either bitmaps or vector graphics. This leads to two different sorts of fonts, *bitmapped* fonts and *outline* fonts, based on these two graphics technologies. Glyphs in bitmapped fonts exhibit the same characteristics as bitmapped images, while outline glyphs share the characteristics of vector graphics. Since character shapes are usually built out of simple curves and strokes,[14] and lack any subtle texture, there is little advantage in using bitmapped fonts on the grounds of expressiveness. The main advantage of bitmapped fonts is that glyphs can be rendered on-screen simply and rapidly, using a bit copying operation. This advantage is being eroded by the wider availability of efficient graphics hardware and faster processors, but bitmapped fonts remain widely available.

14
Indeed, there is a long tradition of using mathematical constructions to define glyphs. A recent example of this approach to font design is the METAFONT system, described in Donald E. Knuth, *The METAFONT Book*, Addison-Wesley, 1986.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 230
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

The considerable disadvantage of bitmapped fonts is that, like any other bitmapped image, they cannot be scaled gracefully. The need for scaling arises in two different contexts: when a font is required at a different size from that at which it was prepared, and when it is required at a different resolution. The latter case is unlikely to arise with multimedia productions designed for display on screen, since the bitmapped fonts installed on computer systems are usually designed for use at screen resolution, but it presents serious problems if text must also be printed on a high resolution printer. The first case — scaling bitmapped fonts to produce type at a different size — always presents a problem. The irregularities that result from scaling a bitmap are especially intrusive with character shapes, owing to the amount of fine detail in their design. When bitmapped fonts are used, it is normal to provide versions of each font at several different sizes. Characters will then look as good as is possible at those sizes, but not at any others, so the designer is forced to use only the standard sizes.[15]

Each platform has its own native bitmapped font format, usually optimized in some way to work efficiently with the system software routines responsible for text display. In contrast, outline fonts are usually stored in a cross-platform format. The two most widely used formats are Adobe *Type 1* (often simply referred to as *PostScript fonts*, although there are other types of PostScript fonts) and *TrueType*.[16]

Outline fonts can be scaled arbitrarily, and the same font can be used for display at low resolution and for printing at high resolution, provided that the display software can interpret the outline shapes. The utility Adobe Type Manager (ATM) performs this function at the system level for Type 1 fonts; rendering of TrueType fonts is built into both the Windows and MacOS operating systems, hence outline fonts of both types can be used by any application program. Because of the supremacy of text preparation software directed towards print-based media, outline fonts are more widely used than bitmaps for elaborately formatted text.

Outline fonts in either of the popular formats can contain up to 256 glyphs. The glyphs in a Type 1 font are simply small programs, written in a restricted subset of PostScript. The restrictions are intended to make it possible for the glyphs to be rendered efficiently on screen by programs such as ATM, at sufficient quality to be acceptable at screen resolution. TrueType is an alternative format, based on quadratic curves instead of the cubic Bézier curves used

15 This is no different from the situation in the days before digital fonts were available when, as we remarked earlier, each size of a typeface in a particular style was a separate font.

16 The TrueType format was originally devised by Apple, but is more widely used on Windows than on MacOS, where, because of the popularity of Adobe software, Type 1 fonts are most common.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 231
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

by PostScript. A character outline in a TrueType font is stored as a series of points which define the lines and curves making up its shape. A newly developed format, *OpenType* unifies Type 1 and TrueType fonts, by providing a file format that allows either sort of outline to be stored.[17]

17 OpenType was developed by Adobe and Microsoft in 1997–8. OpenType fonts began appearing in 1998.

⇨ You might also meet Adobe Type 3 fonts, although these are becoming rare — they date from a period when some aspects of the Type 1 specification were kept secret, and people wishing to develop their own PostScript fonts were obliged to use the inferior Type 3 format. There is also a Type 0 format, in which the elements of the font are not character glyphs, but other fonts. The Type 0 format is intended for storing composite fonts, each built out of several Type 1 fonts. Its main application is in providing fonts with a large character repertoire, such as might be required for use in conjunction with Unicode.

As well as descriptions of the glyph shapes, Type 1 and TrueType fonts both include extra information that can be used by a rendering program to improve the appearance of text at low resolutions. This information concerns rather subtle features, which tend to lose their subtlety when only a few pixels can be used for each stroke. In Type 1 fonts, it takes the form of declarative *hints*, which provide the values of parameters that can be used to direct glyph rendering at low resolutions. For example, some hint information is intended to cause overshoots to be suppressed. As we stated on page 209, overshoots normally help improve the regular appearance of a font. However, at low resolutions, when the amount of overshoot is rounded to the nearest pixel, the effect will be exaggerated and text will appear to be more irregular than it would if all lower-case letters were exactly as high as the x-height. The point at which overshoot suppression is effective depends on the design of the font, so it cannot be applied without information from the font designer. The Type 1 hints allow this information to be supplied. Other hints similarly suppress variations in stem width, which while aesthetically pleasing at high resolutions, merely look uneven at low resolutions. Another example of the use of hints concerns bold fonts. It is quite conceivable that the rounding effect of rendering at low resolution will cause the strokes of a normal font to thicken to such an extent that it becomes indistinguishable from the boldface version of the same font. A hint allows rendering software to add additional width to boldface fonts to ensure that they always appear bold. TrueType fonts achieve the same sort of improvements

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 232
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046



Figure 7.18
Anti-aliased text

using a more procedural approach, allowing the font designer to write *instructions* that specify how features of a character should be mapped to points on a pixel grid at any resolution.

⇨ Although the use of hints and instructions improves the appearance of fonts at small sizes as well as at low resolutions, it only causes minor changes of shape to compensate for the small number of pixels available. Traditional typographers sometimes produce separate designs of the same font at different sizes, which is said to produce a more pleasing appearance than simply scaling one design to fit. Such niceties are rarely encountered with digital fonts, although the Computer Modern fonts, designed to accompany the TEX typesetting system, were produced in this way.

As with other sorts of vector graphics, the appearance of outline fonts can sometimes be improved by anti-aliasing — softening a hard, and inevitably jagged, edge between those pixels that are on and off by using a pattern of grey pixels over a wider area. The smoothing effect thereby achieved is illustrated in Figure 7.18: the upper letter A is not anti-aliased, the lower one is. As this figure shows, anti-aliasing is very effective with large characters; it also works well at medium resolutions, such as that of a laser printer. At low resolutions, small type (below about 12pt), although noticeably smoother, also begins to look blurred, which may leave it harder to read than it would be if left jagged, so anti-aliasing should be applied to fonts judiciously.

Multiple Master Fonts

Before digital typography, each font provided a typeface in one style at one size. With most digital formats, each font provides a typeface in one style, and each size is generated from it by scaling, with the help of hints. Multiple Master fonts are a new development, based on Adobe Type 1 fonts, wherein each font provides an entire typeface, from which any style may be generated at any size.

Multiple Master fonts are conceptually fairly simple. Several characteristics of a font family can be considered as values along a line. For example, the weight of a particular font lies somewhere between ultra-bold and ultra-light (at least, relative to other members of the same family with different weights). A Multiple Master typeface comprises Type 1 fonts for each of the extremities of such a line, or *design axis*, as it is called. Fonts lying between the extremities

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 233
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

can be constructed, as *instances* of the Multiple Master typeface, by interpolation: the weight of glyphs for a medium weight font might lie half-way between the ultra-bold and ultra-light glyphs. Potentially, an infinite number of fonts, differing finely in their weight, can be generated from the two extremities of the weight design axis.[18]

A single design axis is not always very exciting, but design axes can be combined. The Multiple Master format allows the ends of up to four axes to be stored in the same typeface file. Although the format is liberal enough to allow any characteristics to be varied along design axes, four specific ones are registered as part of the specification. They are weight, width (e.g. from condensed to extended), optical size, and serif style. The optical size axis is particularly interesting, since it permits features such as the proportions of letters, their weight and spacing to vary with the point size. This allows a closer approximation to the traditional practice of modifying character shapes to suit the size of type than is provided by Type 1 hints and TrueType instructions (which are primarily concerned with maintaining the quality of rendering). Figure 7.19 shows how three design axes can be combined into a font design space. By labelling each axis with values from 0 to 1, any point within the cube defined by the axes can be identified by three coordinates. The point corresponds to a font instance; the coordinates express the proportions of the characteristics of each design axis relative to its end-points that are present in the instance.

A Multiple Master typeface is larger than a single Type 1 font, since it must include a set of glyphs for each end of each design axis. However, it can be used to generate an entire font family — indeed, a family comprising many more individual font instances than you would normally find in a font family provided as a collection of individual fonts.

Multiple Master typefaces can also be used to provide a partial answer to the problem of font substitution. There are two problems with embedding fonts when you wish to distribute a document. The first, purely practical, is that incorporating fonts into a document may seriously increase its size. Not only does this make the document more difficult to distribute, but the recipient may not wish to accumulate large numbers of fonts, or multiple copies of the same font on their systems. The second problem is that fonts are subject to copyright, and it may not be legal to distribute them freely in a form that allows the recipient to use them for any

18
Although as we are working with finite digital values, the actual number of variations is restricted.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 234
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

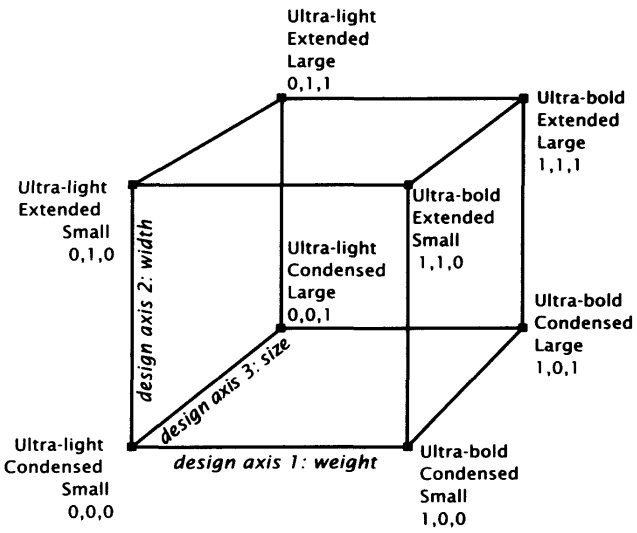


Figure 7.19
Multiple Master design axes

purpose other than to view a particular document. Ensuring that a distributed font can only be used for a single purpose is not always feasible. However, if fonts are not distributed with the document, font substitution will probably occur, destroying the document's carefully designed appearance. Designers wishing to use fonts other than the small number of rather dull fonts that can be found on most computer systems are left in a quandary: to distribute or not?

With Multiple Masters a new solution becomes possible. Instead of distributing the font, include a sufficient description of it to enable an instance of a Multiple Master typeface to be constructed that approximates its appearance and matches its spacing. That way, if a font must be substituted, the deleterious effect on the document's appearance is minimized. This form of font substitution is supported by the *Portable Document Format (PDF)*; the Acrobat Reader software that displays PDF files will instantiate fonts if necessary. A pair of general-purpose Multiple Master typefaces (one serifed, the other sans serif) is distributed with Acrobat Reader, so it is always possible for some approximation to the intended font to be produced. Other Multiple Master typefaces can be obtained, which have distinctive characteristics like a conventional font family.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 235
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Further Information

The layout of text is described in Chapter 8.

Appendix B of [GQ99] is a good account of character sets and related issues; [App93b] includes some interesting information about the diversity of writing systems used around the world.

[Bla92] is a history of fonts in the 20th century; [Cli94] and [HF97] concentrate on recent faces. [Ado90a] is the specification of the Type 1 font format, and [Ado97] is Adobe's guide to Multiple Master fonts.

Exercises

1. In Chapter 3, we described how a graphics application program keeps track of a model of an image, which it renders to make it visible. Discuss the extent to which a sequence of character codes can be considered a model of text, which has to be rendered as glyphs.

2. Naïve commentators sometimes claim that using Unicode or ISO 10646 is wasteful of space, when documents can be adequately represented in ASCII. Explain why this is not so.

3. Although Unicode can represent nearly a million different characters, keyboards cannot have that many keys. Discuss ways in which keyboards can be made to accommodate the full range of characters.

4. Look at a newspaper. Compare and contrast the way fonts are used in news stories and in advertisements.

5. Find out which fonts are available on your computer, and print samples of each. Which would you consider to be suitable for display purposes and which for text?

6. Which of the fonts available on your system would you use for each of the following jobs?

   (a) The opening credits of a science fiction movie.

   (b) Your *curriculum vitae.*

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 236
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

(c) The body copy of a community newsletter.

(d) A wedding invitation.

(e) The list of ingredients on the back of a jar of jam.

Explain each of your choices.

7. Is there a meaningful distinction to be made between fonts that are *readable* and those that are *legible*? If so, what is it?

8. Fonts designed to be used on low-resolution devices usually have a relatively high x-height. Why?

9. Design a font for Scrabble tiles. You only need the upper-case letters. You will have to ensure that your letter forms fit harmoniously and legibly into the square tile (and don't forget the little number for the letter's value). If you don't have access to font design software, such as Fontographer or Metafont, you should design the character shapes on squared paper in sufficient detail to be mapped to pixels; otherwise, use the software available to you to generate a font in a suitable format and use it to typeset some sample words.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 237
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Layout

8

In Chapter 7, we considered the storage and display of individual characters. When characters are combined into words, sentences and extended passages of text, we need to think about how they should be arranged on the screen. How are paragraphs to be broken into lines? How are headings, lists and other textual structures to be identified and laid out? How are different fonts to be used?

As well as being a vital component of multimedia, text can play an additional role by specifying the structure of a multimedia production. Layout commands and visual characteristics can all be specified using a text-based *markup language*. In later chapters we will show how links to other documents, the location of images and video clips, interaction with users, and synchronization of time-based media can also be specified textually in a markup language. The best known such language is HTML, which we will introduce in this chapter, together with the newer and more powerful XML.

Our primary focus is on markup and the way it is expressed in languages, not on the tools, such as word processors and desktop publishing packages, that present users with an environment for preparing text. Multimedia authoring systems usually include similar facilities for preparing text. It is, however, common practice to use a dedicated system to prepare text which is then imported into the authoring environment, already formatted. In either case, the formatting operations available are equivalent to what can be expressed using the languages described later in this chapter.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 238
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Text in Graphics

The maximum flexibility in the layout of text can be obtained by treating text as graphics and manipulating it with a graphics program. Text can then be placed at will, possibly using layers to overlay it, and treated with effects and transformations. The integration of text and graphics occurs naturally, so this approach to text is ideally suited to graphic design incorporating text, such as posters, packaging, company logos and letterheads, Web pages, book jackets, and CD covers. As always, there is a choice between vector and bit-mapped graphics, which determines how the text can be manipulated.

If text items are treated as objects in a vector graphics program, they can be arranged on the page arbitrarily, and all the effects that can be applied to other graphical objects can be applied to them.[1] Words and letters can be scaled, rotated, reflected and sheared. They can be stroked with colours, gradients, or even patterns. The leading drawing programs provide extra facilities for laying out text, which take account of its sequential nature. Tools allow you to fill areas with text, arranged either horizontally or vertically. (Vertical text is intended for languages that are normal written that way, but can be applied to horizontal languages for a special effect.) The text flows automatically to stay within the area, with line breaks being created automatically as required. Text can also be placed along a path.

1 But you must use outline fonts.

Figure 8.1 shows some typical ways of treating text in a vector graphics program. The design down the left was made by setting the name Shakespeare vertically, using a mid-grey to colour the text, taking a copy and reflecting it about the vertical axis. The mirrored copy was then filled with a lighter shade of grey and displaced slightly vertically and horizontally. Glendower's speech is set on a Bézier path, while Hotspur's reply is centred within an oval. Again, slightly different shades have been used to distinguish the components. The attribution has been rotated; all the elements were scaled to fit their spaces. The text remains editable: spelling mistakes can be corrected — you can even use a spelling checker — and the fonts and type size can be changed. The shapes into which the text has flowed, and the paths along which it is placed can be changed using the normal vector editing tools, and the text will move to accommodate the changes. By treating text as bitmapped

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 239
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

graphics, quite different results can be obtained. Once text has been converted into pixels, it becomes susceptible to all the retouching and filtering that painting programs provide. (However, unless the text is set in a large font, many effects either do not show up, or obliterate it entirely.) Text can be warped and distorted, embossed and bevelled, set in burning metal, or weathered and distressed. Figure 8.2 provides some examples. Different elements of the design have been blurred and sharpened; the line 'This is the worst' has been copied onto several layers, displaced relative to each other to produce a 3-D effect, which is enhanced by the streaking that has been painted on. The text 'So long as we can say' is not placed on a path, as it would be in Illustrator, it has been distorted using Photoshop's polar coordinates filter. This text has been incorporated into a bitmapped image and can no longer be edited as text, but must be retouched like any other part of the image. Using the terminolgy from Chapter 7, text in a vector graphics program retains its lexical content, whereas in a bitmapped image it is reduced entirely to appearance.

Evidently, both approaches to text in graphics have their uses, and they can be profitably combined. For most purposes, vector-based text is more manageable; it has been common practice for designers to prepare text in Illustrator, then import it into Photoshop, where it is rasterized so that bitmapped effects can be applied. With the release of Photoshop 5.0, vector-based text was incorporated directly into Photoshop.Text on its own layer can be edited, as it can in Illustrator, although the full range of vector effects is not available. The text must still be rasterized before bitmapped effects can be used. The distinction between the two approaches to graphics and text remains: programs are increasingly combining the two, but the underlying representations remain distinct.

Graphics programs offer complete control over the appearance and placement of text, but this is not always what is required. Quite the contrary, in fact. If you are preparing some conventional text, such as a letter or an article, you want to be able to just type the words, and have a program do as much of the layout as possible for you. For this, we have to turn to a different class of programs, and consider some new questions, in particular, how do we exercise control over layout without having to place everything explicitly?

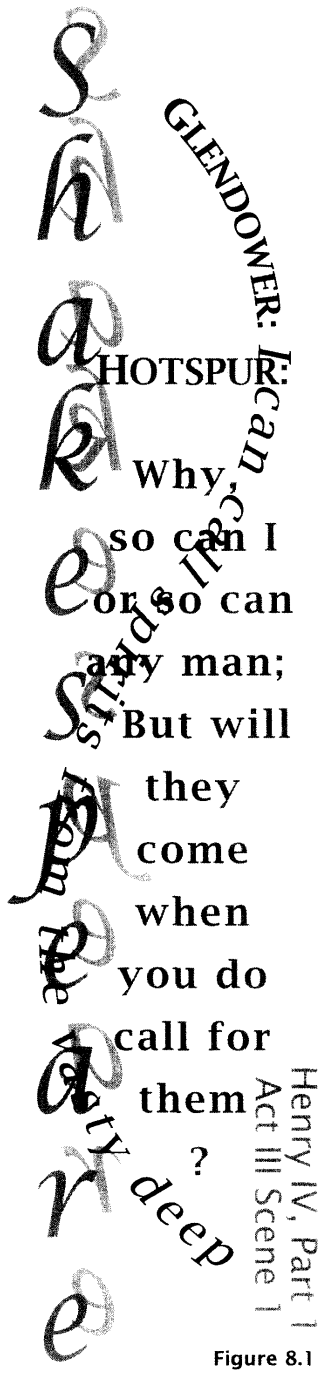


Figure 8.1
Vector text

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 240
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

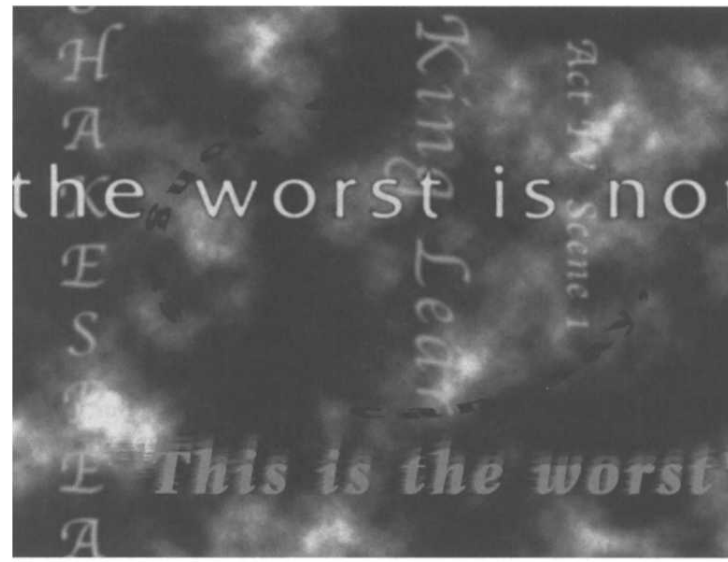


Figure 8.2
Bitmapped text

Markup

In the days before desktop publishing and word-processors, authors' manuscripts were usually produced on a typewriter. Consequently, they could not be laid out in exactly the same form as the final published book or paper. The author, the publisher's copy editor and possibly a book designer would annotate the manuscript, using a variety of different coloured pens and a vocabulary of special symbols, to indicate how the text should be formatted by the typesetter when it was printed. This process of annotation was called *marking up* and the instructions themselves were often referred to as *markup*. With the transition to computer-based methods of publication, the function of markup has been transferred to the instructions inserted (often by the author) into the text file that now corresponds to the manuscript. The form of these instructions is determined by the program being used to prepare the manuscript.

A distinction is often drawn between *WYSIWYG* formatting systems, and *tag-based* systems. WYSIWYG stands for 'What you see is what you get', a phrase which succinctly captures the essence of such systems: as you type, text appears on the screen laid out just as it will be when printed or displayed. Font and size changes,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 241
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

indentation, tabulation, and other layout features are controlled by menus, command keys or toolbar icons, and their effect is seen immediately. The markup, although present, is invisible; only its effect can be seen, giving the illusion that the formatting commands do not insert markup, but actually perform the formatting before your very eyes, as it were. In contrast, with a tag-based system, the document is normally prepared using a plain text editor; the text is interspersed with the markup, which takes the form of special layout commands, often known as *tags*. Tags are lexically distinguished from the text proper, by being enclosed in angle brackets or beginning with a backslash character, for example.[2] Tags do the same job as the commands in a WYSIWYG system, but their effect is not necessarily immediately visible. A separate processing phase is usually required, during which the formatting described by the tags is applied to the text, which is converted into a form that can be displayed or printed, such as PostScript. Word processors such as MS Word are invariably WYSIWYG, as are most commercial page layout tools, such as Quark Express and Adobe inDesign. Tag-based layout has, until recently, been largely confined to the academic and technical sectors, where tools such as `troff` and LaTeX have been in use since long before the phrase 'desktop publishing' was coined. In recent years, the popularity of HTML has brought tag-based text formatting to a wider audience.

[2] A sufficiently powerful editor, such as Emacs or Alpha, will allow you to insert tags with a single keystroke or menu selection.

Although the experience of preparing text using these two different types of system is very different, calling for different skills and appealing to different personalities, a moment's reflection will show you that their differences are mostly superficial and concern the interface much more than the way in which layout is actually controlled. Underneath any WYSIWYG system is a tag-based text formatter, although the tags are more often binary control codes than readable text tags, and are inaccessible to the user of the WYSIWYG editor. The rash of WYSIWYG HTML editors and the re-targeting of page layout packages to produce HTML demonstrate that there is no real incompatibility between using tags to control layout and using a WYSIWYG approach to document preparation.

We will concentrate on layout using textual tags, partly because it is easier to see what is going on, and partly because, after years in the shade of layout based on hidden control codes, it has acquired a new importance because of the World Wide Web. An advantage of using textual markup, instead of binary codes or some data structure, is that plain text can be read on any computer system and

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 242
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

can be transmitted unscathed over a network. This is particularly important for the Internet, with its heterogeneous architectures. Here, HTML has brought tagging into wide use. XML, and a clutch of languages developed from it, promise to extend the use of this sort of markup into other areas of multimedia.

The difference between tag-based and WYSIWYG layout is merely a cosmetic difference between interfaces. A more profound distinction is that between *visual* markup and *structural* markup. In visual markup, tags or commands are used to specify aspects of the appearance of the text, such as fonts and type sizes. In structural markup, tags identify logical elements of a document, such as headings, lists or tables, and the visual appearance of each type of element is specified separately.

The distinction can be illustrated using LaTeX, an example from a previous generation of tag-based markup languages, with which either approach can be used. For example, suppose we were using LaTeX to prepare a document which was divided into sections, each of which had a section heading. If we were taking a visual approach, we might decide that each section heading should be centred and set in large boldface type. This can be accomplished using commands such as the following for each section:

```
\begin{center}
\fontfamily{hlh}\fontseries{b}\fontshape{n}
\fontsize{14}{16}\selectfont Introduction
\end{center}
```

whose meaning should be fairly evident: LaTeX tags begin with a backslash, and some of them take arguments enclosed in curly brackets. Here, `\begin{center}` and `\end{center}` define the extent of an 'environment' within which text is centred. The next, more enigmatic, commands select the font: `hlh` identifies the Lucida Bright family,[3] `b` its boldface series and `n` its normal shape; next we stipulate a size of 14pt on a 16pt baseline, and finally select the font. A cut-and-paste editor, or one that supported keystroke macros, would make it relatively easy to ensure that every section was laid out in the same way.

However, if we were to adopt the structural approach to our markup, we would simply tag each section heading *as* a section heading:

```
\sectionheading{Introduction}
```

3 LaTeX fonts are identified with apparently cryptic names because of the LaTeX community's decision to adopt a naming scheme that could be made consistent with the primitive file naming conventions of MS-DOS when font specifications are mapped to font file names.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 243
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Elsewhere, probably in a separate file, known as a *document style*, containing definitions for all the structural elements of a particular type of document, we would provide a definition of the `\sectionheading` command that specified the layout we wished to apply. For example, if we still wanted each section heading to be centred and set in large boldface type, we could define `\sectionheading` like this:

```
\newcommand{\sectionheading}[1]{
  \begin{center}
    \bfseries\Large #1
  \end{center}
}
```

Again, the meaning should not be too obscure: `\newcommand` defines a new command, whose name appears next in curly brackets. The `[1]` indicates that the command will take one argument, in this case the text of the section heading. The commands between the next pair of curly brackets form the body of the definition; they are substituted in place of `\sectionheading` whenever it is used. The `\bfseries` command chooses a boldface version of the current font and `\Large` switches to a larger point size. (These two commands are themselves abstractions of the primitive font selection commands we used previously. Both of them change the font characteristics relative to the body font of the document, which would be set globally elsewhere.) Within the body, `#1` will be replaced by the argument when the command is used, so `\sectionheading{Introduction}` will lead to `#1` being replaced by `Introduction`. The net effect will be to centre the heading in large boldface type, as it was previously.

Structural markup has distinct advantages over visual markup. Perhaps most obviously, it allows us to change the appearance of a document globally by changing the definitions of markup tags just once. If, for example, we wanted to use a more imaginative layout for our section headings — right-aligned in uppercase boldface italics, say — we would only have to change the definition of `\sectionheading` — for example, to:

```
\newcommand{\sectionheading}[1]{
  \begin{flushright}
    \bfseries\itshape\Large \uppercase{#1}
  \end{flushright}
}
```

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 244
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

and every section would be re-formatted in the new style. Consistency is ensured. In contrast, if we had marked up each section heading individually, we would have had to find and change the markup of each one.

The ease with which structural markup allows layout to be changed offers more than a licence to experiment with different typographical effects at whim. A more substantial advantage is the easy localization of documents which it permits. Centred layouts, for example, which are fairly standard in the United States, are much less popular in Europe. Changing the layout of a document such as a software manual to make it look more agreeable to customers in different parts of the world can be achieved by redefining the heading commands, while leaving the marked up text of the manual alone. Similarly, while an author is working on a manuscript, it might be desirable to preview it using a simple layout, perhaps double-spaced. By using a different set of definitions for the tags, the same markup can be used to produce a final version, laid out in a more professional way.

Even better, a sufficiently powerful system based on structural markup would allow you to redefine the effect of your markup tags for different output media. For example, if you wished to produce a version of your document to be 'read' to blind people by a speech synthesizer, you might redefine a section heading so that the words of the heading were read with increased emphasis.[4] More generally, since the structural tags themselves do not specify any form of processing, they can be used as the basis for presentation of the document in any medium, including media not invented when the document was written.

A related advantage of structural markup is that it permits a separation of concerns, and a division of labour, between the appearance of a document and its structure. The writer need only concentrate on identifying the logical divisions within the document while he or she is writing. The question of what typographical features should be used to identify those divisions in a perspicuous and pleasing way can be dealt with separately, ideally by a specialist book designer.[5] If visual markup is used, the processes of writing and layout become intertwined, and control over appearance is left to the whim of the author.

➲ Programmers will recognize that visual markup is an *abstraction* mechanism, similar to the use of functions and classes in programming. The separation of appearance from markup resembles the separation of a class's interface from its implementation.

4
This has actually been done for LaTeX. See *An Audio View of LaTeX Documents - Part II* by T.V. Raman, in TUGboat, 16,3, September 1995.

5
This is the situation that prevailed in the publishing industry prior to the arrival of 'desktop publishing'.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 245
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

The final advantage of structural markup is that tags identifying the structural elements make it easy for a computer program to analyse the structure of the document. Although people can identify the structural elements from their appearance, this depends on knowledge and understanding, things which it is notoriously difficult to incorporate into computer programs. Programs are extremely good at searching text for strings with a specific form, such as LaTeX commands, though. This makes it easy to write programs to extract tables of contents or other tagged pieces of information. As a simple example, the Alpha editor is able to construct a pop-up menu of all the section headings and sub-headings in a LaTeX document — provided they are marked up using the `\section` and `\subsection` commands, and not an *ad hoc* sequence of visual markup — so that, when a section name is selected from the menu, the insertion point is moved to the start of that section. In a more complex environment, structural tags make it possible to display an outline of a document, at varying levels of detail, or to allow readers to search for strings only when they are used in a particular way, as part of a heading or a figure caption, for example.

We must emphasize that, although we have used LaTeX in our examples, and will continue to use tag-based markup in the remainder of this chapter, structural markup can be applied using modern WYSIWYG word-processors and page layout programs. The mechanisms and their names vary, but macros, stylesheets, and master pages all offer the possibility of separating the appearance of a document from its structure.

SGML

In the preceding section we gave examples of both structural and visual markup using LaTeX. If you are familiar with LaTeX, you will know that it was designed specifically as a structural markup system built on top of the low-level formatting features provided by the powerful typesetting system TeX. Despite this, in LaTeX we could choose visual formatting, because the same language is used for markup of the text and for defining new commands. To perform the latter function it must provide facilities for visual formatting, and there is nothing to stop you using these visual facilities to mark up your text.

There is much to be said for providing computer users with the choice between different ways of working, but many of the benefits

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 246
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

of structural markup can only be obtained if it is used consistently and exclusively, and the only way to guarantee this happening is by completely separating markup from visual layout. The easiest way to enforce this separation is via a language that provides nothing but facilities for structural markup. This philosophy finds its purest embodiment in *SGML (The Standard Generalized Markup Language).*

SGML is arguably mis-named, since, properly speaking, it is not a markup language, but a markup *metalanguage.* That is, it is a language for defining markup languages; it allows you to define your own tags, each of which is used to mark a particular sort of *document element*, such as a paragraph, section heading, or list. In SGML, defining a tag consists of giving it a name and some attributes, and specifying some restrictions on the context in which the tag may be used and *nothing more.* This is in contrast to LaTeX and similar systems, where the only way to define a new tag is as a command, whose definition specifies the layout to be applied when it is used, so that a LaTeX document class defines the structure *and* the appearance of a class of documents. The SGML equivalent, known as a *Document Type Definition (DTD)* defines only the structure. In other words, a DTD describes all the documents of a particular type, in terms of the tags that may be used to mark them up. A document marked up with SGML tags from a DTD thus contains a description of its own structure. Any program that needs to understand the document's structure can deduce it by reading the tags.

➲ If you are familiar with programming language definitions, you may find it helpful to consider a DTD as being like an attribute grammar defining a class of documents, in a similar way to that in which a grammar defines the syntax of a class of programs — and thereby defines a language.

We are used to thinking of a document as a physical object, and of its content as what appears on the paper, but SGML's notion of a document goes beyond the limitations of physical media, to encompass more of the potential of computer-based storage. A document may include elements which are not intended to be displayed, but which describe its contents in some way. For example, a document may include a tag which indicates the language in which its text content is written. This information is largely redundant for display — it will soon become evident to the reader — but might be valuable to a document retrieval

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 247
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

system, allowing it to select only documents in a language that could be understood by the person doing the retrieving. The SGML tag would distinguish the language information from the other elements of the document. SGML can thus be used not just for traditional text markup, but for marking up any sort of data as a self-describing document. Remember, though, that in order for this to be possible, a DTD defining the class of documents in question must be available: in the example just cited, the language tag can only be included in documents based on a DTD which includes it.

From a structural markup perspective, displaying is only one of the things that might be done to a document, but from our multimedia perspective, it is the most natural, so we must consider mechanisms for controlling the layout of a structurally marked-up document. SGML provides no such mechanism, because of its determination to separate the description of a document's structure from its appearance. But, precisely because of this, it permits the use of *any* mechanism based on structural information. Several approaches are possible.

If a DTD describes a sufficiently large class of documents, it may make sense to construct display software specifically for that class of documents, and to hardwire the layout rules into the software. That way, every document will be laid out consistently by a particular display program. A variation on this approach is to build facilities into the display program allowing its user to control some aspects of the layout. This is the approach adopted by the first generations of Web browsers[6] (and remains the default in more recent ones). Most of the formatting of headers, lists, tables and other elements is controlled by the browser, but some details, specifically fonts, type sizes and colours, may be controlled by preferences set by the user. This is quite a radical departure from traditional publishing, since it gives the reader control over some aspects of the appearance of the document.

The alternative is to provide a separate specification of the layout, parallel to and complementing the DTD. Such a layout specification is usually called a *stylesheet*. For each tag defined in the DTD, a stylesheet provides a rule describing the way in which elements with that tag should be laid out. There may be more than one stylesheet for a particular DTD, providing a different appearance to the same structure, as, for example, one corresponding to typographical styles favoured in the US, and another corresponding to those in Western Europe. Stylesheets are written using a *stylesheet language*, which is designed for the purpose.

6 Web pages may be described by an SGML DTD — it's called HTML.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 248
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Ideally, the use of stylesheets in conjunction with SGML leads to scenarios such as the following. A technical publisher devises a DTD for books, defining tags for each element that might occur in a technical book, such as chapters, sections, paragraphs, quotations, tables, computer program listings, footnotes, and so on, hopefully exhaustively. Authors mark up their manuscripts with the tags defined in this DTD. For each book, or series, a book designer constructs a stylesheet, which has the effect of laying out the book in a style appropriate to its contents and intended market, taking account of any budget constraints and house style rules which might affect the layout.

This ideal scenario is not always achieved, however, not least because desktop publishing software has accustomed authors to having control over the appearance of their work. In practice, SGML has been used in the publishing industry mainly as an archiving format for electronic versions of printed material, since its independence from any formatting mechanism or technology means that the markup will in principle remain valid forever and can be used as the basis for publication at any time in the future, using whatever technology has become available. Furthermore, the markup can be used for indexing and document retrieval, thereby making the archived material readily accessible. However, SGML never remotely approached the status of a universal markup language, despite being enshrined in an international standard.[7]

7
ISO 8879:1986(E). *Information Processing — Text and Office Systems — Standard Generalized Markup Language (SGML).*

One reason for this was the lack of any perceived need for structural markup. This perception changed with the increased use of heterogeneous computer networks, such as Arpanet and subsequently the Internet. Since one can make no assumptions about what hardware or software might be on the other end of a network connection, visual markup that is tied to one piece of software or depends on certain hardware facilities for correct display cannot be relied upon to be intelligible or effective. Since structural markup can be interpreted on any platform in a logically consistent — if not necessarily visually consistent — manner, it is ideally suited for the markup of documents which are to be transmitted over networks.

⇨ At one time, the only type of text that could be reliably transmitted over wide area networks and be guaranteed to be readable on any computer was plain ASCII, a situation that led to many important documents, such as Internet RFCs, being stored on archives with no more formatting than could have been achieved on a typewriter.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 249
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

When the World Wide Web was being designed, therefore, the structural approach was adopted for the markup of Web pages. SGML was used as the basis of the *Hypertext Markup Language (HTML)* devised for this purpose. Unlike SGML, HTML is only a markup language: it provides a set of tags suitable for marking up Web pages, but does not provide any way of defining new tags. In fact, HTML is defined by an SGML DTD,[8] which describes the structure of the class of documents known as Web pages. Like SGML, it is concerned solely with identifying the structural elements of a page, and not with their appearance.

8 The original version of HTML was not, but all versions subsequent to HTML 2.0 have been.

Originally, the World Wide Web was intended as a means of dissemination of scientific research, so the DTD for Web pages contained tags corresponding to the main elements of a scientific paper: a title, headings that can be nested several levels deep to introduce sections, subsections, subsubsections and so on, and lists of various types such as enumerations and bullet points. Additionally, HTML provides some typographical control, by allowing you to mark text as italicized or bold, although it does not allow you to specify particular fonts;[9] HTML also provides generic styling tags, to mark text as, for example, emphasized or strong, without specifying anything about the typographical rendering of such text — these tags are more in the spirit of structural markup than are the explicit typographical tags. Later revisions of HTML added tables and interactive forms to its repertoire. As well as these layout tags, HTML crucially includes tags for hypertext linkage and for inclusion of images and other media elements, as we will describe in later chapters.

9 Font selection was introduced in HTML 3.2, having been provided as an extension in popular browsers previously, but is deprecated in HTML 4.0 in favour of stylesheets.

XML

Although they are adequate for marking up the simple, mainly textual, papers for which they were originally intended, HTML's layout tags are by no means sufficient for all the diverse types of material that have found their way onto the World Wide Web, particularly since its commercialization during the middle of the 1990s. During that period two undesirable responses to HTML's deficiencies were made by browser manufacturers and Web page designers. The first response, from the manufacturers, was the addition of *ad hoc* proprietary extensions to HTML. Usually such additions were only supported by one browser, thus destroying the platform-independence of HTML markup, and often they were not

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 250
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

in the spirit of structural markup.[10] The second response, from designers, was to put the available tags to purposes for which they were never intended. The most widespread form of HTML abuse — ensconced in more than one HTML authoring program and encouraged by any number of HTML tutorials — is the use of tables for grid-based layout. This trick enables designers to use established design ideas from traditional media to produce more interesting and readable page layouts than the simple sequence of paragraphs that HTML provides.

10 The most blatant example is perhaps the <BLINK> tag.

Another trick that has been widely adopted is to use graphics instead of text for parts of a document that cannot be satisfactorily formatted with HTML. Mathematical equations provide an example: there are no facilities in HTML for typesetting mathematics, so the only satisfactory way to incorporate equations and formulae into a Web page is to typeset them using some system, such as LaTeX, that can handle them, and then transform the typeset result into a GIF image that is pasted into the Web page. The same approach has been used by designers who are deeply concerned about the appearance of their pages to make sure that text is set in exactly the font they envisaged, and laid out just as they wished it to be.

Even if you believe that designers should live with HTML layout when they are working on the Web, there are many cases where the information they have to present simply does not fit in to an HTML document. Consider, for example, a restaurant wishing to post its menu on a Web page. What tags should be used for each of the dishes on offer? Probably, using a level 2 or 3 header would produce appropriate formatting, but dishes from a menu are not headings, any more than blocks of text laid out on a grid are entries in a table, or mathematical equations are illustrations. If structural markup does not describe the structure of a document, but rather another structure that happens to look like what the designer had in mind as the visual appearance of the document, then much of the point of structural markup is lost. In particular, automatic searching and indexing programs cannot correctly identify the elements of the document. For example, it would be hard to write a program to search for restaurants that served chocolate mousse unless it was possible to distinguish occurrences of the phrase 'chocolate mousse' that announced its presence on a menu from all others, such as headings in on-line recipe collections or studies on nutrition.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 251
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Since HTML's tags are not sufficient for the class of Web pages as it has evolved, new tags are required. Any attempt to add tags to HTML to accommodate absolutely any sort document that someone might one day want to make into a Web page is doomed to failure, and would leave HTML bloated and unmanageable. A much better solution is to provide Web designers with a facility for defining their own tags. This is where we came in: SGML has just that facility. SGML is not, however, entirely suitable for use over the Internet; certain of its features make parsing it too difficult to be done efficiently enough to provide the required response times. Work on adapting SGML to the Internet led to the definition of a subset, known as *XML (eXtensible Markup Language)*, which provides all the important facilities of SGML without the overhead imposed by full SGML. In particular, XML allows Web designers to define their own DTDs for any type of document. Web pages are thus freed from the limitations of HTML's definition of a document.

XML 1.0 was adopted as a W^{3}C Recommendation early in 1998. Because of its complexity compared with HTML, it is unlikely that XML will be used directly to mark up individual Web pages by many designers. It is more likely that XML will be used indirectly: experts will produce XML DTDs that define new markup languages for specific tasks. Although this will lead to a proliferation of special-purpose languages, they will all share some syntactic features, such as the form of tags, and will all be based on the underlying technology of XML, so that it will not be necessary to add interpreters for each new language to every browser. So long as it can interpret XML DTDs, a browser can interpret pages marked up in any language defined by one. XML can thus function as an infrastructure supporting a wide variety of mechanisms for organizing text and multimedia content. There is no reason for it to be restricted to the World Wide Web in this rôle; XML can function equally well with other delivery media.

The process of defining new languages as XML DTDs began before XML 1.0 was even adopted as a Recommendation. Among the DTDs produced are *SMIL (Synchronized Multimedia Integration Language)*, *PGML (Precision Graphics Markup Language)*,[11] *MathML (Math Markup Language)* and *XFDL (eXtensible Forms Description Language)*.[12] A somewhat different application of XML is as a concrete syntax for *RDF (Resource Description Framework)*, which is intended to provide a standard for *metadata* — data about a document, such as its language, or labels describing its content for the benefit of filtering services.

11 Now superseded by SVG, also defined by an XML DTD.

12 HTML is defined by an *SGML* DTD, not an XML one, since it uses some of the features of SGML which were dropped in the XML subset. Work is proceeding on an XML DTD for HTML, which will bring it into line with these other markup languages.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 252
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Although we have introduced the idea of structural markup in the context of text layout, the ideas behind it may be applied to other media and to their integration into multimedia. We will show in later chapters how XML is used in some of the languages just listed to manipulate richer media than pure text. HTML, too, has features for combining other media with text, and these will be introduced when we come to consider multimedia proper.

Text Layout Using HTML and CSS

Our main interest in describing HTML is to introduce you to the syntax of markup in languages of the SGML family, and to illustrate structural markup. Similarly, with CSS we intend to show how stylesheets can provide sophisticated layout control to complement the markup. We assume that most readers will have some knowledge of HTML, but will not necessarily be acquainted with CSS. Those readers needing additional detailed information should consult the suggestions for further reading at the end of this chapter.

HTML is a language that has evolved — not always in an entirely controlled manner — and its design goals have changed over the years. As a result it displays inconsistencies, especially between its current structural markup ethos and left-over visual markup features from an earlier phase of its evolution. Thus, whereas some tags are little more than hooks on which to hang stylesheet-based formatting, others provide direct control over typography. Because of the need for browsers to be able to display old Web pages, HTML must provide backwards compatibility with its older versions, so this mixture of features will persist. Looking at this situation in a positive light, HTML will continue to support a plurality of approaches to markup.

⇨ The features of HTML described in this book correspond to the HTML 4.0 specification, adopted as a W^{3}C Recommendation in December 1997. Some of them are not supported by earlier versions of HTML. Similarly, we include some features from the CSS2 Recommendation of May 1998 which are not present in CSS1.

Although we have referred to Web pages being displayed by a Web browser, such as Netscape Navigator or Internet Explorer, the structural markup of HTML can also be interpreted by other

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 253
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

programs, such as a text-to-speech converter. In the following pages, we will adopt the name *user agent*, used in the HTML specification, to mean any program that interprets HTML markup. A user agent, such as a conventional Web browser, that formats HTML documents and displays them on a monitor, is an example of a *visual user agent*; a text-to-speech converter is an example of a non-visual user agent.

Elements, Tags, Attributes and Rules

HTML markup divides a document into *elements*, corresponding to its logical divisions, such as sections and paragraphs. In general, elements may contain other elements: a section typically contains several paragraphs, for example, and this containment is reflected in the markup. Each element is introduced by a *start tag* and ends with an *end tag*; between these two tags is the element's *content*, which, as we just observed, may include other elements with their own start and end tags. Elements must be properly nested, that is, if an element starts inside another, it must end inside it, too. This means that each element forms part of the content of a unique *parent* element that immediately encloses it.

Every type of element available in HTML has a name. Its start tag consists of the name enclosed in angle brackets, as, for example, <P>, which is the start tag of the P element, used for paragraphs. The end tag is similar, except that the element's name is preceded by a slash: the end tag for paragraphs is </P>. So every paragraph in an HTML document has the form:[13]

13
But see page 238.

```
<P>
    content of the paragraph element
</P>
```

All other elements have the same form — start tag, content, end tag — but with their own identifying tags. The case in which an element name is written in a tag is ignored: <P> and <p> are both the same tag and begin a P element. The case in an end tag may be different from that in the start tag it is paired with.

Some other lexical details have to be considered. Prior to the HTML 4.0 specification, ISO 8859-1 was the designated character set for HTML. In HTML 4.0, the character set is ISO 10646; no encoding is specified, you are allowed to choose your own, provided you specify it with the appropriate MIME type, and the HTML

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 254
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

specification defines 'encoding' liberally, to encompass subsetting. Hence, most HTML documents continue to be written in the ISO 8859-1 subset of Unicode. As true Unicode support begins to be provided by user agents, this situation should change.

An inevitable problem arises if tags are written in the same character set as the text proper: we must lexically distinguish tags from text, using some special characters (in HTML the angle brackets that surround the element name in a tag) so we need some mechanism for representing those special characters when they appear in the text as themselves. In HTML, a < symbol can be represented by the *character entity reference* `<` (the terminating semi-colon is part of the entity reference). This now leaves us with the problem of representing an & so as not to cause confusion. Again, a character entity reference is used; this time it is `&`. Strictly speaking, there should be no trouble using > symbols in text — there is always sufficient context to determine whether or not it is part of a tag — but some user agents do get confused by them, so it is advisable to use `>` except at the end of a tag.

⇨ As you may have guessed, character entity references provide a general mechanism for inserting characters that are hard to type or are not available in the chosen character encoding. References are available for all the characters in the top half of ISO 8859-1, as well as for mathematical symbols and Greek letters, and for layout characters, such as an em-dash. Some examples are given in Table 8.1; for the full list, consult section 24 of the HTML 4.0 specification. There are many Unicode characters without character entity references. For these you can use a *numeric character reference*, which specifies a character using its ISO 10646 character code. A character with code *D* is written as `&#`*D*`;`, so, for example, `<` is an alternative to `<`. (If you are more comfortable with base 16 numbers, you can write your numeric entity references in hexadecimal, putting an x after the # to indicate you are doing so. Another alternative to `<` is `<`.[14])

Table 8.1
A few examples of character entity references

`¡`	¡
`£`	£
`§`	§
`Å`	Å
`å`	å
`ϖ`	ϖ
`∀`	∀
`∃`	∃
`∈`	∈
`∋`	∋
`—`	—
`†`	†
`‡`	‡

14
Or `&#X3C` — case is not significant.

Whitespace consists of space, tab and formfeed characters, and line breaks, which may be of the form used on any of the major platforms: carriage return, linefeed or both. In text, whitespace separates words in the usual way. Visual user agents will format the text in accordance with the conventions of the writing system being used; in English this will usually mean that runs of spaces are replaced by a single space. (In PRE elements, described later, whitespace is left exactly as it is typed.) Whitespace immediately following a start tag or preceding an end tag is ignored, so that the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 255
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

following two examples will be displayed identically.

```
<P>
Shall I compare thee to a summer's day?
</P>
<P>Shall I compare thee to a summer's day?</P>
```

An HTML document may be annotated with *comments*. These are introduced by the character sequence <!-- and terminated by -->. For example,

```
<!-- This section was revised on
     21st September 1998
-->
```

As the example shows, comments may occupy more than one line. Comments are not displayed with the rest of an HTML document. They are intended for information about the document, for the benefit of its author or anyone who needs to read the marked up source. However, as we will see later, comments have also been appropriated for the purpose of maintaining compatibility between current and older versions of HTML, by providing a mechanism for hiding content that older browsers cannot deal with.

⇨ As if the syntax for comments was not odd enough already, HTML allows you to leave space between the -- and the closing > (but not between the opening <! and its --). As a result, HTML comments are very difficult to parse correctly, and most browsers don't really try. Hence, it is very unwise to include strings of hyphens within an HTML comment, and you cannot rely on comments nesting correctly.

In HTML you are allowed to omit end tags where the context makes it clear that they should be there. For example, most documents will consist of a series of paragraphs, which should be of the form:

```
<P>
   content of the first paragraph
</P>
<P>
   content of the second paragraph
</P>
<P>
   content of the third paragraph
</P>
```

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 256
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

and so on. Even a computer program can tell that if you are starting a new paragraph, the previous one must have ended, so the </P> tags can be left out:

```
<P>
   content of the first paragraph
<P>
   content of the second paragraph
<P>
   content of the third paragraph
```

Any good book on HTML will tell you which end tags can be omitted and when. There is no particular merit in doing so, unless you actually type all the tags yourself instead of using an intelligent editor or HTML authoring application. In our examples we will show all end tags.[15]

15 The omission of end tags is an example of *markup minimization*, a feature of SGML that has been carried over into HTML but is not present in XML. This is one of the reasons that HTML cannot easily be defined by an XML DTD.

There is one case in which end tags *must* be omitted, and that is the case of *empty elements*. These elements have no content. An example is the HR element, which produces a horizontal rule (line). There is no sensible content that can be supplied to such an element — it just *is* a horizontal rule — so there is nothing for an end tag to do and you must leave it out.

Although it does not have any content, a horizontal rule element has some properties, including its width and colour. In HTML[16] such properties are called *attributes*. The attributes associated with a particular element are specified as part of its definition in the DTD. Values for each attribute may be assigned within the start tag of an element. For example, to set the length of a rule to one half of the width of the displayed page and its width to 12pt, we would use the following tag:

16 And SGML and XML.

```
<HR size = "12" width = "50%">
```

The case of attribute names is ignored; we use lower case to emphasize the distinction from element names. Values are assigned as shown using an = sign. Any number of assignments may occur inside the tag, separated by spaces. Attribute values are written enclosed in either single or double quotes. (If you need a double quote inside an attribute value enclosed in them, use the character entity reference ".) The treatment of Boolean attributes (which may take the value true or false) is singular: there are no Boolean constants, and a Boolean attribute is set to true by assigning its own name to it, as in:

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 257
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

```
<HR size = "12" width = "50%" noshade = "noshade">
```

(The noshade attribute, when true, causes the rule to be displayed as a solid block, and not as the pseudo-3D groove that is normally used.) The notation for setting Booleans is clumsy, so a contracted form is allowed, where the mere presence of the attribute is sufficient to set it to true. The previous example would more normally be written:

```
<HR size = "12" width = "50%" noshade>
```

Booleans are set to false by omitting them from the tag, as in the first 12pt rule example.[17]

17
User agents commonly recognize the values `true` and `false`, although these are not part of the HTML 4.0 standard.

HTML experts might cringe at the preceding examples, because all the attributes to HR that we have used are *deprecated* in HTML 4.0. Actually, most of the attributes of simple text elements are deprecated, because they control some aspect of appearance. This can now be controlled much better using stylesheets. In principle, since stylesheet information is separate from markup, any stylesheet language can be used with HTML. In practice, most Web browsers only support *Cascading Style Sheets (CSS)*, a simple stylesheet language which works well with HTML, and can be easily mastered.

Let's return to paragraphs to demonstrate how a stylesheet can be used to control layout. For now, do not worry about how the stylesheet and HTML document are combined.

A paragraph element is a logical division of the text of a document. In the absence of any stylesheet, a user agent is free to express this division in whatever way it chooses. In English text, several different conventions are employed for laying out paragraphs. The visually simplest relies only on the insertion of additional vertical space between paragraphs; commonly this visual cue is augmented or replaced by applying an indentation to the first line of each paragraph. Most, if not all, visual user agents use the simplest convention by default. Suppose we want to use the more elaborate form, with indentation.

CSS allows us to specify various visual properties of each document element. One of these properties is, luckily for our example, the indentation of the first line. We specify that the first line of each paragraph should be indented by 4pc with a CSS rule like this:

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 258
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

```
P {
    text-indent: 4pc;
}
```

The rule has two parts: a *selector* (here P) which indicates which elements the rule applies to, and some *declarations* (here there is only one) which provide values for some visual properties. Here the declaration specifies a `text-indent` (the name of the property controlling the amount of indentation applied to the first line) of 4pc. The declaration part of a rule is enclosed in curly brackets, as you can see.[18]

18
If you have ever written an Awk script, this syntax should look familiar.

Whenever the rule just given is in force, every paragraph will be displayed with its first line indented by any user agent that implements CSS stylesheets (and is configured to do so). Suppose now that we did not wish to apply this indentation to every paragraph. Many manuals of style suggest that the first paragraph of a section should not be indented, for example. In order to be more selective about which paragraphs a rule applies to, we need some way of distinguishing between different classes of paragraph, such as those that are indented and those that are not. The HTML attribute `class` can be used for this purpose. This attribute is a property of virtually every HTML element; its value is a distinguishing name that identifies a subset of that element. For example, we might use a class `noindent` for paragraphs we wished to be displayed with no indentation.

In a CSS rule, a selector can consist of an element name followed by a dot and a class name. To specify that paragraphs of class `noindent` should not be indented, we would add the following rule to the previous one:

```
P.noindent {
    text-indent: 0pc;
}
```

As you would probably expect, when there is a general rule for some element, like our first example, and a more specific rule for some class of that element, like our second example, the more specific rule is applied to elements of that class, and the general rule is only applied to elements belonging to no class, or to some class for which no specific rule is available. If, for example, we has some paragraphs of class `unindent`, they would be displayed with a 4pc first line indent, since the only rule that applies to them is the general one with selector P.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 259
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

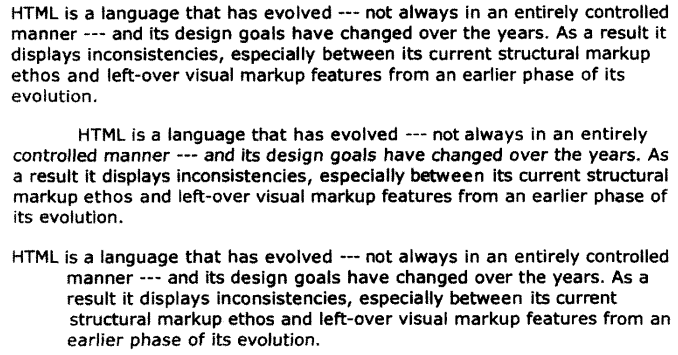


Figure 8.3
Browser display of paragraphs with different indents

Rules can be used to control a range of properties; even a simple rule with only a couple of declarations can produce effects that are not usually available in HTML. For example, the following rule causes paragraphs to be displayed with a hanging indent.

```
P.hang {
    text-indent: -40pt;
    margin-left: 4pc;
}
```

The effect of these three rules is shown in Figure 8.3, which shows the display of the HTML in Figure 8.4 by a user agent that understands stylesheets.

Figure 8.4 also illustrates some features of HTML that have not been described yet. The very first line is a *document type declaration*, which identifies the DTD defining the version of HTML being used. The enigmatic form of the DTD specification should just be treated as red tape; the declaration is not actually needed by current browsers, but it can be used by verification programs to check the legitimacy of the HTML markup. Next comes the start tag for the HTML element. This is the root of the entire document structure; all the elements of the document proper are contained within it. There are only two document elements that can come immediately inside the HTML element: HEAD followed by BODY. The head of the document contains information about the document, which is not displayed within it; the body contains the real text.

There are three elements contained in the head of the document in Figure 8.4. The first META element should also be treated as red tape: it specifies that the character set is ISO 8859-1. This is the default

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 260
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
                        Transitional//EN">

<HTML>
<HEAD>

<META http-equiv="content-type"
      content="text/html;charset=iso-8859-1">
<TITLE>Paragraphs</TITLE>

<STYLE type="text/css">
<!-- /* Hide content from old browsers */
P {
text-indent: 4pc;
}
P.noindent {
text-indent: 0pc;
}
P.hang {
text-indent: -40pt;
margin-left: 4pc;
}
/* end hiding content from old browsers */ -->
</STYLE>
/HEAD>

<BODY>
<P class="noindent">
HTML is a language that has evolved
--- not always in an entirely controlled manner --- and
its design goals have changed over the years.  As a
result it displays inconsistencies, especially between
its current structural markup ethos and left-over
visual markup features from an earlier phase of its
evolution.
</P>
<P class="unindent">
HTML is a language ...
</P>
<P class="hang">
HTML is a language ....
</P>
</BODY>
</HTML>
```

Figure 8.4
HTML source for Figure 8.3

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 261
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

usually applied, but it is good practice to specify the character set explicitly in this way. If you do so, the META element should be the first thing in the HEAD (for reasons that are hopefully obvious).[19] The TITLE element contains a short title, which is often displayed in the title bar of the browser's window. It is not displayed within the window, because it is not part of the text contained in the body.

19 If your chosen character set does not provide ASCII as a subset you have got problems.

The next element in the HEAD is a STYLE, which is where the stylesheet rules governing the layout of this page are to be found. (This is not the only place they can be, we will see an alternative later.) The start tag has an attribute `type` whose value is the MIME type of the stylesheet. For CSS rules, this is always `text/css`. The next line is a trick we will see more than once. The usual reaction of a Web browser to tags it does not understand is to ignore them completely: the tags are skipped, so the content would be displayed as if it were text. This is sensible behaviour much of the time, but quite inappropriate for stylesheets. The syntax of CSS allows an entire set of rules to be enclosed in an SGML/HTML comment, which has the effect of hiding it from browsers that do not recognise the <STYLE> tag. The remarks on the same lines as the HTML comment tags are CSS comments, which indicate what is going on.[20]

20 HTML editors and authoring programs often insert these comments automatically.

The body of the document contains the paragraphs themselves, with their class attributes. The style in the head is applied to them to produce the desired display.

To summarize: text in the body of an HTML document is marked up with tags that delineate the document elements corresponding to the logical divisions of the text. Stylesheet rules, which may be put in a STYLE element in the document's head, control the layout of these elements; the selector of each rule determines which elements it applies to, and the corresponding declarations set properties that control the appearance on screen. The `class` attribute is used to distinguish between different subsets of an element type, so that finer control can be exerted over their layout.

We are now in a position to describe the HTML tags that can be used to mark up text, and the CSS properties that can be used to control its layout. Although we are describing these particular languages, you should appreciate that the underlying principles of markup and layout apply to any system of text preparation.

Please note that the account we are about to give is not exhaustive. Some additional details will be provided later — in particular, we will look at facilities for incorporating other media besides text

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 262
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

in later chapters — but we are not attempting to give a definitive reference guide to or tutorial on HTML and CSS. For that, consult the specifications or suggestions for further reading at the end of the chapter.

HTML Elements and Attributes

The HTML 4.0 specification defines 91 elements, of which 10 are deprecated, meaning that they may be removed from future revisions of HTML, since there are now preferred ways of achieving the same effect. Only a few of these elements are concerned purely with text layout. Those that are can conveniently be divided into *block-level* and *inline* elements. Block-level elements are those, such as paragraphs, that are normally formatted as discrete blocks; i.e. their start and end are marked by line breaks. Inline elements do not cause such breaks; they are run in to the surrounding text.

The most frequently used block-level textual element is the paragraph (P) element, which we have looked at already. Other block-level elements concerned purely with text layout include level 1 to level 6 headers, with element names H1, H2...H6; BR, which causes a line break; and HR, the horizontal rule element which we saw previously. The BLOCKQUOTE element is used for long quotations, which are normally displayed as indented paragraphs. Note, though, that using BLOCKQUOTE as a way of producing an indented paragraph is an example of the sort of structural markup abuse that should be avoided: markup is not intended to control layout. This being so, you will appreciate that the PRE element, which is used for 'pre-formatted' text and causes its content to be displayed exactly as it is laid out, is something of an anomaly, yet it is useful for cases where the other available elements do not serve, and elaborate stylesheet formatting is not worthwhile.

The only elaborate structures that HTML supports as block-level elements are lists and tables. Tables are relatively complex constructions (as they inevitably must be to accommodate the range of layouts commonly used for tabulation); since their use is somewhat specialized we omit any detailed description. Lists, in contrast, are quite simple. HTML provides three types:[21] ordered lists, in the form of OL elements, unordered lists, UL elements, and definition lists, DL elements. OL and UL elements both contain a sequence of list items (LI elements), which are laid out appropriately, usually as separate blocks with hanging indentation. The difference is that

21 Two other special-purpose lists are also available, but both are deprecated.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 263
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

- first item, but not numbered 1;
- second item, but not numbered 2;
- the third item contains a list, this time a numbered one:
    1. first numbered sub-item;
    2. second numbered sub-item;
    3. third numbered sub-item;
- fourth item, but not numbered 4;

ONE
    the first cardinal number;
TWO
    the second cardinal number;
THREE
    the third cardinal number

Figure 8.5
Browser display of HTML lists

```
<UL>
    <LI>first item, but not numbered 1;
    <LI>second item, but not numbered 2;
    <LI>the third item contains a list, this time a numbered one:

    <OL>
        <LI>first numbered sub-item;
        <LI>second numbered sub-item;
        <LI>third numbered sub-item;
    </OL>
    <LI>fourth item, but not numbered 4;
</UL>

<DL>
    <DT>ONE <DD>the first cardinal number;
    <DT>TWO <DD>the second cardinal number;
    <DT>THREE   <DD>the third cardinal number
</DL>
```

Figure 8.6
HTML source for Figure 8.5

user agents will number the items in an ordered list; the items in an unordered list are marked by some suitable character, often a bullet. The items of a DL element are somewhat different, in that each consists of two elements: a term (DT) and a definition (DD). The intended use of a DL is, as its name suggests, to set lists of definitions; typically each item consists of a term being defined, which will often be exdented, followed by its definition. Figures 8.5 and 8.6 show the use of lists. Note that a list item element can contain a list, giving nested lists.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 264
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

The most abstract block-level element is DIV, which simply identifies a division within a document that is to be treated as a unit. Usually, a division is to be formatted in some special way. The `class` attribute is used to identify types of division, and a stylesheet can be used to apply formatting to everything that falls within any division belonging to that class. We will see some examples in the following sections. Even in the absence of a stylesheet, classes of divisions can be used to express the organizational structure of a document.

⇨ Although we emphasize the use of the `class` attribute and stylesheets for formatting, HTML provides most textual elements with attributes that provide some primitive layout control. For example, the paragraph element has an attribute `align`, which may take the values `left`, `center`, `right` or `justify`, causing the paragraph to be left-aligned (ragged right), centred, right-aligned (ragged left) or justified (fitted flush to both margins by adjusting the inter-word spacing). The `align` attribute may also be used with HR elements and all levels of heading. When it is used with DIV, it causes all the block-level elements contained within the division to be aligned according to its argument. For example,

```
<DIV align="center">
<P>  first paragraph </P>
<P>  second paragraph </P>
<P align="left">  third paragraph </P>
<P>  fourth paragraph </P>
</DIV>
```

would cause all but the third paragraph to be centred. The third paragraph would be left-aligned, because explicit alignment of an element within a division overrides that of the DIV. Although stylesheets provide superior formatting facilities, it may be necessary to use attributes such as `align` so that user agents that do not support stylesheets can produce an approximation to the intended layout.

Inline elements are used to specify formatting of phrases within a block-level element. As such, they might be seen as being in conflict with the intention of structural mark-up. It is, however, possible to identify certain phrases as having special significance that should be expressed in their appearance without compromising the principle of separating structure from appearance. Examples of elements that work in this way are EM, for emphasis, and STRONG for strong emphasis. Often the content of these elements will be displayed by a visual user agent as italicized and bold text, respectively, but

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 265
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

they need not be. In contrast, the I and B elements explicitly specify italic and bold text. These elements *are* incompatible with structural markup and should be avoided. (Especially since a stylesheet can be used to change their effect.)

There is an inline equivalent to DIV: a SPAN element identifies a sequence of inline text that should be treated in some special way. In conjunction with the class attribute, SPAN can be used to apply arbitrary formatting to text, as we will see.

In this section we have considered textual elements that can appear in the body element of an HTML document, but there is one element that can *replace* the body, and that is the FRAMESET. Framesets are interesting because they provide a type of layout that, while deriving from traditional practice, exploits the unique characteristics of screen displays. A frameset divides a page up into a collection of individual *frames*, arranged in a rectangular grid. To that extent, it is just an analogue of the grid often used in magazine layout. However, HTML frames can be updated independently (in ways which will be described later), which opens up new possibilities. Typically, some frames are kept constant, displaying headlines, advertisements, navigational information or copyrights and attributions, for example, while others are used to display changing content. This general form of layout has become something of a cliché on commercial Web sites, but at least it is a cliché based on the form of the medium, and not one transplanted from some other medium.

The FRAMESET element has among its attributes rows and columns, which allow you to specify the arrangement of the grid. Several possibilities exist, including using absolute sizes, but the easiest option is to specify the width of the columns and the height of the rows as percentages of the size of the whole page. For example,

```
<FRAMESET rows="80%,20%" cols="20%,80%">
```

specifies a grid of four frames, arranged in two rows and two columns, with the first row occupying 80% of the height of the page, and the first column 20% of its width. As the example demonstrates, the values of both the rows and cols attributes are strings describing the size of each row and column; the number of rows and columns is implicit, commas are used to separate the individual parts of each attribute value. If either rows or cols is omitted, its value defaults to "100%".

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 266
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

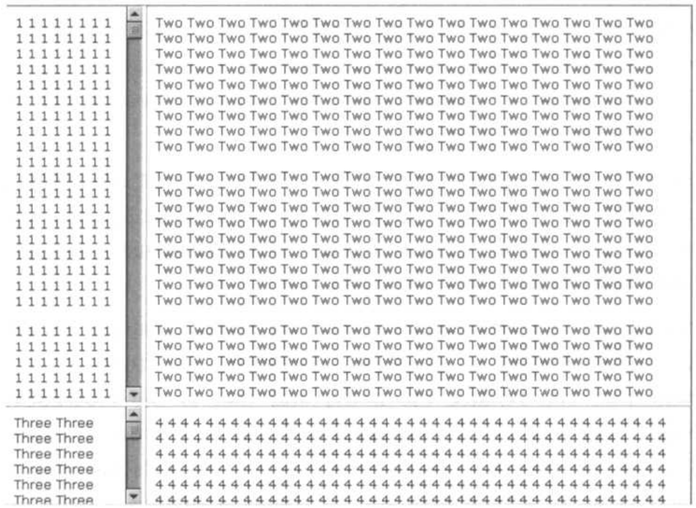


Figure 8.7
Browser display of a simple frameset

A **FRAMESET** element can only contain **FRAME** elements or other framesets. Each frame is (perhaps unexpectedly) an empty element; it uses an attribute named `src` ('source') to specify another HTML file, whose contents will be displayed within the frame. The value of the `src` attribute is a URL.[22] The **FRAME** elements are fitted into the frames specified by their parent frameset in left-to-right, top-to-bottom order, as shown in Figure 8.7, which was produced by the HTML in Figure 8.8. These figures also show how the `scrolling` attribute can be used to add scrollbars to individual frames. (Note also in Figure 8.8 that the document type for a document using frames is different from that which we saw previously.) Nested framesets can be used to produce more complicated frame layouts.

22
See Chapter 9 if you have managed to avoid knowing what a URL is.

⇨ Framesets must be used judiciously. There is ample anecdotal evidence to suggest that many users find them confusing, precisely because the elements of a page become independent of each other, which undermines the metaphor of a printed page that is generally used to make sense of Web browsing.

All the elements we have described can possess a `class` attribute, which permits subsetting. Additionally, each may have an `id` attribute, which is used to specify a unique identifier for a particular occurrence of the element. For example,

```
<P id="para1">
```

is the start tag of a paragraph identified as `para1`. This identifier can

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 267
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
<TITLE>Frames Example</TITLE>
</HEAD>

<FRAMESET rows="80%,20%" cols="20%,80%">
<FRAME src="frame1.html">
<FRAME src="frame2.html" scrolling="no">
<FRAME src="frame3.html">
<FRAME src="frame4.html" scrolling="no">
</FRAMESET>
</HTML>
```

Figure 8.8
HTML source for Figure 8.7

be used in various ways, one of which is in a CSS selector, where it must be prefixed by a # symbol instead of a dot. For example,

`P#para1` { *formatting to apply to this paragraph only* }

CSS Properties

It is now time to see how CSS can be used to transform the sparse text markup provided in HTML into an expressive and flexible formatting system. Again, you should note that the principles of layout — the values that can be changed and the ways they can be combined, for example — apply to many text preparation systems. Master pages in a DTP system such as FrameMaker, document styles in LaTeX, and even macros in a word-processor perform a very similar function to that of stylesheets and are applied to the text of a document via its markup in a similar way. In turn, these formatting operations closely resemble the tasks that have been performed in setting type by hand for hundreds of years.

It should be emphasized that CSS is a language for visual formatting, so it can only be meaningfully interpreted by visual user agents. Non-visual user agents have their own stylesheet languages.

CSS allows you to control the typography of your document, by choosing fonts and setting the type size. Five properties control the font characteristics described in Chapter 7. Several of them display some ingenuity in coping with the inevitable uncertainty about the capabilities of user agents and the availability of fonts. This is most

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 268
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

evident in the `font-family` property: its value may be a list of font names (separated by commas) in decreasing order of preference. For example:

```
P.elegant  { font-family: "The Sans", Verdana,
                          Helvetica, sans-serif }
```

says, in effect, that we would ideally like text in paragraphs of class `elegant` to be set in a rather recherché font called 'The Sans'. (Note that we must surround the name by double quotes in the CSS declaration because it contains a space.) Should that font not be available, we will settle for Verdana, failing which Helvetica will do. If even Helvetica is not possible, any sans serif font should be used. Our fallback choices are based on pragmatic considerations: Verdana is similar to the preferred font, and is distributed with Internet Explorer, so there is a good chance that it will be available. Next we try for a sans serif, which, although not as good a match, is almost certain to be on any user's system. Finally, we have used a generic font family, in the expectation that a user agent will substitute an appropriate font that falls into that family. CSS provides five such generic families: `serif`, `sans-serif`, `monospace`, `cursive` and `fantasy`. The first four correspond to font styles introduced in Chapter 7; the fantasy family is a catch-all for display fonts, experimental fonts and general eccentricity. The actual font selected when a generic font must be used will depend on the configuration of the browser. There is no actual guarantee that it will fall into the class identified by the name: the standard set of fonts included on a MacOS system does not include any font that would qualify as a fantasy, for example, so something more down-to-earth would have to be used.

⇨ The naming of fonts is not standardized, so you must be careful to include in a list of font family names all the pseudonyms by which your chosen font is known. For example, Times New Roman is often identified simply as Times or Times-Roman. If you wanted to be as sure as possible that this font would be used you would have to include all three names at the beginning of your font family list.

The search for a font is carried out for each character — remember that HTML uses ISO 10646 as its character set. Where a document contains text in several languages, or mixes mathematical symbols with text, the glyphs for different characters will be found in different fonts. The list of fonts in a declaration for the `font-family` property can be used to group together a set of fonts that

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 269
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

should be used to provide glyphs for all the characters that might occur in a document.

The two properties `font-style` and `font-variant` are used to select different font shapes. The `font-style` property can have the values `normal`, `italic`, or `oblique`, `normal` being upright, and `oblique` what we have termed 'slanted'. The value of `font-variant` may be `normal` or `small-caps` — CSS considers small-caps a variant form rather than a different style, in theory allowing for small-caps italic fonts, and so on. The effect of declarations for these properties is to select an appropriate member of the font family chosen on the basis of the value of `font-family`. A slanted font is considered to be an appropriate choice for the font style `italic` if no real italic font is available.

The `font-weight` property has to deal with the fact that the terms used for font weights only make sense within a font family; as we remarked in Chapter 7, the bold version of some typeface may well be lighter than the medium version of another. For simple situations, CSS lets you use the values `normal` and `bold` for this property, with the expected effect. However, many font families provide more than two different weights, but with no universal naming convention to distinguish between them. Instead of imposing a single naming scheme, CSS uses numbers to identify different weights. You can set `font-weight` to any of the nine values 100, 200, ..., 900, which represent an ordered sequence of fonts, such that, as the `font-weight` value increases (numerically), the font's weight will not decrease, and may increase. Thus, if there are nine different weights (TrueType fonts always have nine different weights), each value will select a different one, but if the font has fewer, some values will be mapped to the same weight of font. Finally, you can use the values `bolder` and `lighter`. This requires a slight digression on the subject of inheritance.

The formatting described by a CSS rule's declaration is applied to any document element that matches the rule's selector. As you know, each element has a parent element. Any properties that are not explicitly changed by a rule for an element are left with the values they had in the parent — the properties' values are *inherited*. This is almost certainly what you would expect. It introduces the possibility of specifying property values not in absolute terms, but relative to the inherited values. Values for `font-weight` of `bolder` and `lighter` are the first example of this that we have seen. Their effect is to set the font weight to the next larger or smaller numerical

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 270
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

value that corresponds to a different font from the inherited one, if such a font exists. (You cannot, for example, make a font any bolder than 900.)

A similar option is available for the `font-size` property. This may take on the values `smaller` or `larger`, which cause a relative size change. Font sizes can also be specified as a percentage of the parent element's font size, or as a multiple of the em or ex of the inherited font. Sizes can also be specified independently of the parent font. Here the range of values can be chosen from `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large` and `xx-large`; it is suggested that these values correspond to sizes forming a geometric progression, with a ratio of 1.5. The absolute values will be determined by the choice of size for `medium`, which will be the 'natural' size of the font, as determined by the user agent. There is no guarantee that all user agents will produce the same set of sizes, even for the same font. Sizes can also be specified as absolute lengths, in any unit, although points will usually be preferred.

Normally, when we select a type size, we also specify the line spacing — the vertical distance between baselines. In CSS, the `line-height` property is used for this purpose. The default value, `normal`, allows the user agent to select a 'reasonable' size. This is invariably too small: most user agents (following the advice in the CSS specification) choose a value between 1.0 and 1.2 times the font size. This follows convention for printed matter. This convention is partly dictated by economic considerations: if the lines are closer together, it is possible to fit more words on a page. This consideration is irrelevant for screen display, where the more open appearance lent to text by more widely spaced lines can compensate for the physical strain most people experience when reading from a screen. A line height of about 1.5 times the font size is more suitable. This can be set with any of the following declarations, which illustrate some of the different values available for this property.

```
line-height: 150%;
line-height: 1.5;
line-height: 1.5em;
```

Line heights may be expressed as a percentage, a ratio or in units of ems, all of which are relative to the font in use in the current element. Heights may also be specified as absolute values in any units.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 271
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

All of the font properties may be combined in a shorthand declaration for the `font` property. Its value is a list comprising the values for the five font properties; the individual property values are separated by spaces, leaving enough context to identify a list of font styles separated by commas.[23] A value for `line-height` can be combined with the `font-size`, separating the two with a slash. This closely follows printers' conventions: 12pt/14pt is the way a printer specifies a 12pt font on 14pt baselines. A typical font declaration would look like:

```
P  {  font: italic bold 14pt/21pt
           "The Sans", Verdana, Helvetica, sans-serif
   }
```

Where no attribute is supplied for a font property (such as `font-variant` here) the default is used. This is usually the value inherited from the parent element.

⇨ The value for `font` can also be one of the names `caption`, `icon`, `menu`, `messagebox`, `smallcaption` and `statusbar`, which refer to the font, in all its aspects, used by the host system to label components of its standard user interface: `messagebox` is the font used in system dialogue boxes, for example. These fonts can be used if you wish to harmonize the appearance of your document with that of the host system's desktop.

One of the important differences between printing and displaying text is that printing in colour is an expensive process, but displaying in colour is free — although you cannot guarantee that everyone has a colour monitor. Therefore, using coloured text and text on coloured backgrounds is an option which should be considered and is often to be preferred to imitating ink on paper by using black text on a white background. This is because the white areas of a screen are actually emitting light, which is usually quite bright and, as we noted in Chapter 6 has a very high colour temperature; it can therefore be quite a strain to look at for long — it is as if the monitor was shining a bright light in your eyes. Furthermore, the high contrast between black text and a white background can result in an optical illusion, whereby the type appears thinner, as if the background had spread over it. The same effect occurs in reverse if white text is placed on a black background. A combination of pale grey text on a dark grey background is surprisingly effective: the dark background reduces the strain caused by the light, and the lessened contrast reduces or eliminates the apparent distortion.[24]

23 It should not matter what order the values appear in, but browsers are sometimes picky: style, variant, weight, size, family is a safe choice.

24 The mass protest over the Communications Decency Act, when many Web pages' backgrounds were switched to black, improved the appearance of the Web overnight.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 272
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

In some circumstances, the use of coloured type can serve other ends. For example, if you are making a multimedia production for small children or primary school teachers, the use of primary colours for the text may make it appear more cheerful and inviting. Red on black may produce a more sinister effect, suitable for a digital Fanzine for a Goth band, perhaps. Many style guides will tell you to be restrained in your use of colour, especially bright colours and harsh contrasts, and your own experience probably tells you that over use of coloured text can be initially irritating and ultimately boring. But a timid and conservative approach to colour is boring too.

The two CSS properties `background-color` and `color` control the colour of the background and the text, respectively. Their values specify colours in the sRGB colour space (see Chapter 6), although user agents are given considerable latitude in how faithfully they approximate sRGB colours. Several formats are provided for colour specifications. The most intuitive takes the form `rgb(`*r*`%, `*g*`%, `*b*`%)`, where *r*, *g* and *b* are the percentages of red, green and blue in the desired colour. Instead of percentages, you may use numbers in the range 0–255 if you prefer. The most commonly used form of colour specification is the least readable: the three components are expressed in hexadecimal (base 16), and combined into a single, six-digit number, preceded by a `#`. The following specifications all describe the same shade of mauve: `rgb(80%, 40%, 80%)`, `rgb(204, 102, 204)` and `#CC66CC`.

⇨ If, as in this example, each of the component values consists of a pair of repeated digits, the specification can be abbreviated to a three digit form, with each being given just once, as in `#C6C`. If the digits are taken only from the set 0, 3, 6, 9, C and F, the resulting colour will necessarily belong to the 216 colours of the 'Web-safe' palette.

You can also use names for some colours, but the only ones sanctioned by the standard are the sixteen belonging to the VGA colour palette, so this option is of limited use.

Applying colour, fonts and line spacing to different document elements, such as the body text, headers, emphasized passages and general `DIV`s and `SPAN`s goes a long way to producing attractive and expressive text for use in multimedia. CSS — and DTP packages — provide control over several more aspects of a document's appearance. We will briefly describe some of the more useful of these, leaving the details to the references given at the end of this chapter.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 273
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Lines of text may be aligned in four different ways. In *left-aligned* (or 'ranged left') text, the left margin is straight and words are added to each line until there is not enough room left for another. At that point a new line is started. Since lines will not all be the same length, this results in a ragged right margin. *Right-aligned* text is the same in principle, but the right margin is kept straight by inserting extra space at the left of each line, so that it is the left margin that looks ragged. *Justified* text keeps both margins straight by inserting extra space between words, while *centred* text allows both margins to be ragged, arranging lines symmetrically beween them. In all cases, words might be hyphenated to prevent excessively uneven lines in aligned or centred text or inter-word spacing that is too wide in justified text.

In printed books, justified text is the rule, although left-alignment has many advocates, including respected book designers; it has been enjoying one of its periodic upsurges in popularity during the 1990s. Fully justified text has a neat and orderly appearance, whereas ragged right is less formal, some say more restless. However, even if you wish to present a formal appearance, justification is best avoided for text to be displayed on a monitor. Because of the low resolution of monitors, text is usually set in relatively large sizes, which means that fewer words can be fitted on to a line of reasonable length; this in turn provides fewer gaps between which to distribute the extra space needed for justification, with the result that the text may acquire large or uneven gaps, rivers of white, and other undesirable artefacts that interfere with readability. Right alignment and centring are usually reserved for headings and similar displayed material, although occassionally setting entire paragraphs in one of these modes can be jolly.

Control over alignment in CSS is provided by the `text-align` property, which takes values `left`, `right`, `center` or `justify`. For example, to set the body text of a document ragged right (i.e. left-aligned), except for paragraphs of class `display`, which should be centred, the following rules could be used.

```
BODY      {  text-align: left   }
P.display {  text-align: center  }
```

Text layout for multimedia has more in common with magazine and advertising layout than with mainstream book design. Rather than trying to construct a layout for extended reading of flowing text, the multimedia designer is concerned with arranging text on a

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 274
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

screen, so that each screenful stands on its own and communicates its message or makes its impact as effectively as possible. An established approach to such layout is to place individual blocks of text on a grid. Designers have been using HTML tables as grids, positioning each block in a table cell, but CSS provides direct control over the positioning of document elements, a facility comparable to the control offered by page layout programs such as PageMaker. To understand positioning in CSS you need to know about CSS's model of text rendering, which is typical of the models employed by layout programs, although less sophisticated than, for example, that of TEX.

The model is fundamentally simple, although the fine detail (which we will gloss over) gets quite involved in places. You will recall that HTML document elements can be classified as either block-level or inline. The layout algorithm reflects this classification. Each element is notionally placed into a box. Text that is not contained in any inline element is placed into an anonymous box. These anonymous boxes and those containing inline elements are placed next to each other horizontally, and then this line of boxes is, as it were, folded up to fit into the available width. The alignment specified for the parent element is used to control the distribution of space between the boxes. In this folding process, some boxes may be split between adjacent lines; for example, an EM element may need to be broken across lines. When the inline elements and text contained in a block-level element have been arranged as described, they are placed in a box corresponding to the whole block. The boxes for blocks are placed vertically, one below another.[25] For displays, there is no notion of 'available height' — windows scroll — so no further adjustment or selection of page breaks is necessary.

25 TEXperts will recognize vertical and horizontal mode.

In the absence of any further stylesheet information, this layout algorithm produces the familiar sequence of uniformly spaced blocks of text, all fitted into lines of the same width. CSS lets you exert some control over the disposition of text, to produce more interesting and effective layouts.

⇨ Lists are treated slightly differently, since we usually want to display the label or marker for each list element outside the block containing the element itself. The modification to the basic algorithm is simple. A somewhat more complicated algorithm is used to lay out tables, although in all cases, layout is performed by arranging boxes next to each other horizontally and vertically. Full details can be found in the CSS specification.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 275
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

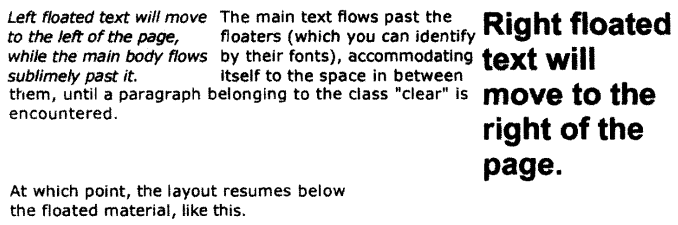


Figure 8.9
Floating text

Each box can be surrounded by a *border*, which is separated from the box's contents by some *padding*; beyond the border, *margins* can be used to separate the box from its neighbours, or from the edges of its enclosing box. Each box's width and height can be specified explicitly, subject to certain constraints. Colour and backgound can be specified separately for each box. These properties permit such effects as shaded and outlined boxes, narrowed columns and superimposed or overlapping elements (using negative margins). Borders, margins and padding are controlled by a plethora of CSS properties, which provide control over each of them on each of the four sides of the box.

It is also possible to make boxes float to the left or right margin, while text flows around them. This facility is most often used for embedding images into paragraphs of text, but it can also be used, in conjunction with margin settings, to float text past text, as shown in Figure 8.9, produced from Figure 8.10. The `float` property can take the values `left` or `right`, with the expected effect. A complementary property `clear` is used to control the placement of text that might be flowing past a floated box. It takes values `left`, `right`, `both` or `none`, indicating which sides of the box may not be adjacent to a floating box. Putting it another way, a value of `left` for `clear` forces a box to go below the bottom of any left-floating element, and so on.

The ultimate in layout control comes from using *absolute positioning* of elements. If an element is formatted according to a rule that sets the property `position` to `absolute`, then you can assign lengths to the properties `top` and `left`, thus determining the position of the top left hand corner of the box. In conjunction with `width` and `height` these properties allow you to place boxes arbitrarily on the screen. This, of course, allows you to place boxes on top of each other, which raises the question of their stacking order. The `z-order` property can be used to control this: its value

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 276
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

```
<HTML>
<HEAD>
<TITLE>Box effects</TITLE>
<STYLE TYPE="text/css">
P.leftfloater {
   margin-left: 0;
   float: left;
   width: 30%;
   font: italic;
}
P.rightfloater {
   margin-right: 0;
   float: right;
   width: 30%;
   font: bold x-large sans-serif;
}
P.clear {
   clear: both;
   width: 50%;
}
</STYLE>
</HEAD>

<BODY>
<P class="leftfloater">
Left floated text will move ...
<P class="Rightfloater">
Right floated text will move ...
<P class=boxed>
The main text flows ...
<P class="clear">
At which point,...
</BODY>
</HTML>
```

Figure 8.10
HTML and CSS source for Figure 8.9

is a number; elements with higher `z-order` values are placed in front of those with lower values. Use of the absolute positioning properties allows you to lay out text with the same degree of control over its positioning as you could achieve using a graphics program, as described in the first section of this chapter, but without having to convert to a graphics format, and without losing the text content or its markup.

The calculations and measurements needed for absolute positioning can be tedious, so most designers will prefer to use software that enables them to manipulate text boxes interactively on their screens, and generates the markup and stylesheet automatically. Such

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 277
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

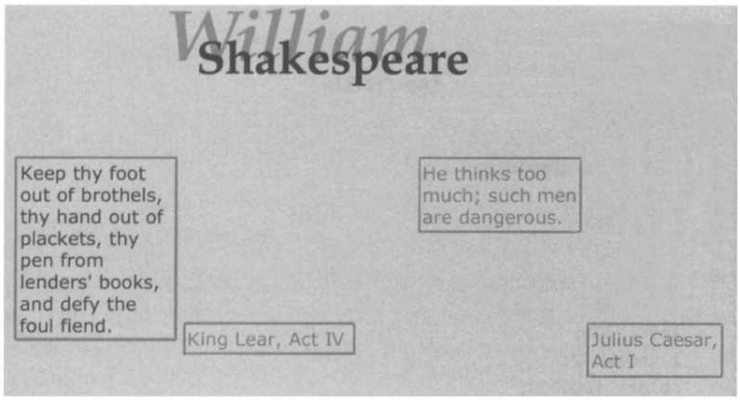


Figure 8.11
Absolutely positioned text

programs essentially provide the layout capabilities of conventional page layout packages.

Figure 8.11, produced from Figure 8.12 is a simple demonstration of absoute positioning.

Several words of caution is in order. You should never forget that user agents may ignore (or fail to correctly interpret) stylesheets.[26] It can be quite difficult to ensure that a document that is laid out using absolute positioning remains comprehensible when it is laid out using a user agent's default interpretation of the structural markup alone. The effect of positioning directives interacts with font sizes, which may be overridden by user preferences. You should also remember that your documents may be rendered by a non-visual user agent, perhaps for the benefit of people with impaired vision. Here again you should avoid producing documents whose comprehensibility depends on their layout.

26
Absolute positioning is also one of the areas where fourth generation Web browsers are most buggy.

In our examples, we have shown stylesheets embedded in HTML documents. Where a stylesheet is to be applied to more than one document, duplicating it in every one is wasteful and leads to maintenance problems. Under these circumstances, a stylesheet is usually stored in its own file, and incorporated into every HTML document that needs it, by way of a `LINK` element, which for this purpose, has the form:

```
<LINK href=" stylesheet's URL"
      rel="stylesheet" type="text/css">
```

(We will describe `LINK` elements more fully in Chapter 9.)

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 278
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

```
<HTML>
<HEAD>
<TITLE>Absolutely Positioning</TITLE>
<STYLE TYPE="text/css">
H1 {
    position: absolute;
    top: 0%;
    left: 25%
}
H1.over {
    top: 5%;
    left: 30%;
    z-index: 2;
    color: #000000;
    font: normal bold 36pt palatino, times, serif;
}
H1.under {
    z-index: 1;
    color: #808080;
    font: normal italic 54pt palatino, times, serif;
}
P  {
    border-color: red;
    border-width: thin
}
P.q1 {
    position: absolute;
    width: 125pt;
    height: auto;
    left: 10pt;
    top: 120pt
}
P.src1 {
    position: absolute;
    left: 140pt;
    top: 250pt
}
```

Figure 8.12
HTML and CSS source for Figure 8.11

Our intention in presenting this description of CSS is to illustrate the sort of control over text layout that is available to designers, by examining in some detail one particularly accessible means of achieving it. We have by no means described all its features, and have simplified some of those we have discussed. Two features in particular that are worthy of note in passing are context-sensitive selectors and media types. The first of these allows you to stipulate that a declaration should only be applied to elements that occur within another specified element type. For example, you might

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 279
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

```
P.q2 {
    position: absolute;
    width: 125pt;
    height: auto;
    left: 320pt;
    top: 120pt;
}
P.src2 {
    position: absolute;
    left: 450pt;
    top: 250pt;
}
BODY {
    color: #AAAAAAA;
    background: #FFFFCC;
}
</STYLE>
</HEAD>
<BODY>
<H1 CLASS="over">Shakespeare</H1>
<H1 CLASS="under">William</H1>
<P CLASS="q1">
Keep thy foot ...
</P>
<P CLASS="src1">
King Lear, Act IV
</P>
<P CLASS="q2">
He thinks ...
</P>
<P CLASS="src2">
Julius Caesar, Act I
</P>
</BODY>
</HTML>
```

Figure 8.12
HTML and CSS source for Figure 8.11 (continued)

wish to apply a special sort of formatting to `EM` elements that occur within other `EM` elements, or within paragraphs belonging to the class `special`, say. Media types allow you to specify that certain parts of a stylesheet apply only when the document is being rendered to certain media. Thus, you can specify different fonts for printing on paper and for displaying on screen, for example.

⇨ You may be wondering why *Cascading* Style Sheets. Stylesheets may be provided by document designers, but they may also be provided by the user; user agents must necessarily provide a default stylesheet. The term 'cascading' refers to the way in

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 280
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

which stylesheets from these sources are combined. Basically, designers' stylesheets supplement or override the users', which in turn supplement or override the user agent's. A user's stylesheet may, however, designate certain rules as *important*, in which case they override any conflicting rules in the designer's stylesheet. A user with poor eyesight might, for example, use an important rule to ensure that text is always set in a large size. However, designers' rules can also be important, and in that case, override any users' rules. This is a facility that should be used with discretion — it is, at the least, impolite to insist on imposing your vision on all users, even though you are a sensitive artist, when you don't know what special needs they might have.

Portable Documents

We have argued in favour of structural markup, but we must now concede that there are occasions when it is not adequate. The reasons for this can be found in two inherent limitations of the model of markup and display derived from SGML.

The first of these is the delegation to the user agent of the task of combining markup with stylesheets. What this means is that the designer cannot be sure what is going to happen when his or her marked up text is eventually displayed. Users might override parts of a stylesheet with important rules of their own; a user agent might ignore stylesheet information, or make a poor job of interpreting it; resizing a window may cause a user agent to reflow text, completely changing the layout of a page. Without knowing which user agent is being employed and how it is configured, you cannot know what the displayed text will finally look like. If you have been convinced by the arguments in favour of structural markup, you will retort that that is precisely the point. If, however, you have seen a carefully devised layout turned into an unrecognizable travesty by a Web browser that cannot interpret the stylesheet that describes it, you may infer that the point of structural markup is not entirely one in its favour.

The second limitation of structural markup concerns the finer points of layout. It is widely accepted among typographers that what distinguishes really fine layout is not the marks but the spaces. The adjustment of inter-word spacing on a justified line to produce a visually uniform appearance is an operation that depends on an

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 281
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

appreciation of how the shapes of the adjacent characters affects their apparent spacing. A clearly visible example is the need to add extra space when the font changes from a slanted or italic shape to an upright shape, unless punctutation immediately follows the italics. Compare this *good* example with this *bad* example. Notice how, in the latter, the ascender of the *d* seems to lean over too far towards the following word. Failure to make this *italic correction* looks particularly bad at low resolution on screen; Web browsers do not make it, and sometimes the result is almost illegible as the inter-word space disappears under the overhang of an italic letter. Other similar niceties that form part of the traditional typesetter's craft include correct hyphenation, the choice of the ratio of the inter-word spacing to the line length, and the spacing after punctuation. Although we have come a long way from word processors that could only use monospaced fonts, performed justification by inserting fixed-width spaces, and hyphenated 'God', only the most classy document preparation and page layout packages provide a level of control over these details approaching that of the best hand typesetting, and in most cases, some user intervention in the form of visual markup is required to produce the best results. This markup cannot be applied to structural elements because it is determined by the way lines are made up, which is a feature that structural markup abstracts away. Therefore, if you use structural markup, you must entrust these fine details of layout to user agents. Contemporary (fourth generation) Web browsers make a very poor effort at this task. In principle, much better results can be achieved automatically, but in practice even such sophisticated spacing and hyphenation algorithms as those employed by TEX cannot match the results that can be produced by hand.

Often in multimedia production, these considerations will be irrelevant. Low resolution displays preclude the most elegant spacing, and it will often be possible to design text components in such a way that their precise layout is not important. Sometimes, though, only the best will do, or a particular layout may be an integral component of the way information is being presented. What can be done then?

Achieving the desired appearance is not the problem, provided one is prepared to live with visual markup and the consequent loss of flexibility and abstraction. Distributing the resulting document is not so easy. To confine ourselves to pure text for the moment, let us suppose that we have prepared a document using LATEX, and

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 282
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

hand-tweaked it to perfection. If we want to send it to you, what can we do? We could send you the marked-up source. This will be no use to you at all unless you have up-to-date versions of TEX and LATEX, and all the macro packages we might have used, installed on your computer. Even if you do, we cannot rely on your having the fonts we have used, unless we restrict ourselves to a very limited choice; we cannot send you our fonts, because that is probably illegal. We have the same problems if, instead of sending the source, we send the device-independent output produced by TEX, since you still need a TEX back-end driver to process it (even though you will no longer need all the LATEX packages). We can translate this output into PostScript and hope that you can interpret that. This is a popular option, but retains some difficulties. Chief among these is the necessity for you to have a PostScript interpreter of some sort available. PostScript printers remain more expensive than other types, and although free software interpreters, such as GhostScript, are available, which can also render PostScript to the screen, they are slow and clumsy. We still have our usual problem with fonts: either we must hope that you share our taste in fonts, or must embed fonts in the PostScript, thereby increasing its size considerably. In desperation, we might convert the output to GIF or some other widely used bitmapped graphics format, but, apart from the inevitably substantial size of the resulting graphics files, this would require us to choose a resolution, which might not be appropriate.

Although we have used LATEX as our example here, much the same situation prevails with any other page layout program: either the author has to hope that the reader possesses the same software and fonts, or the document must be distributed as PostScript or some other graphics format. Multimedia authoring packages such as Director do not provide a solution to this dilemna, since their own built-in text handling facilities are no more sophisticated than a typical word-processor and do not offer the high-quality typesetting we need. Only by importing the output from a high-end page layout program as graphics can we incorporate properly laid out text in a Director movie.

What is needed is an *portable format* for typeset text, and preferably one that can accommodate other elements of multimedia too. Adobe's *Portable Document Format (PDF)* is an attempt to provide exactly that. PDF can be thought of as a simplified PostScript, with which it shares its imaging model. That is, PDF allows shapes to

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 283
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

be described in vector terms, and supports the precise placement of characters from outline fonts. This means that a PDF file can accurately represent the effect of any markup that has been applied, so that the document will be displayed exactly as it was intended to be.

PDF also provides a solution to the problem of fonts. The preferred way of dealing with the fonts in a document is to embed in the PDF file not the font itself but a description of it, consisting of its name, character widths and sufficient other font metric information to characterize the font's general appearance. If the named font is present when the PDF document is displayed, it will be used; if it is not, then the font metrics will be used to construct an instance of a Multiple Master font with the same character widths (so that layout is not disturbed) and similar font characteristics, so that the appearance of the page — the density of the print and its lightness or darkness — closely approximates the effect that would be produced by the specified font. If this is not considered acceptable, Type 1 fonts can be physically embedded in PDF. (Symbol and dingbat fonts, in fact, must be embedded, because font substitution only works for text fonts.)

Although it was originally designed as a text and graphics format, in later releases, PDF acquired facilities for hypertext links, inclusion of sound and video, and integration with the World Wide Web. It can now be seen as a platform-independent multimedia format, based on visual layout.

But, like any other 'portable' format, PDF is only any use to someone with a program that can display it. Adobe overcame this problem by the sublimely simple strategy of producing such a program — Acrobat Reader — for all major platforms and giving it away free.[27] They also published the specification of PDF so that anyone can write a PDF interpreter. The Acrobat Reader package includes the two basic Multiple Master fonts (serif and sans serif) so that the font substitution mechanism is also assured to work.

All of which is all very well, but how is PDF to be generated? There are two mechanisms. On MacOS and Windows systems, where printing is performed as a system-level function, the PDFWriter program can be used as if it was a printer driver: if it is selected for printing in the same way as any real printer, any program that produces printed output can produce PDF simply by sending it to PDFWriter, using the same command as would be used for printing.[28] Where this mechanism is not appropriate, and on Unix

27 We have lost count of the number of copies of Acrobat Reader we have received on CD-ROMs and as part of the distribution of other programs. You probably have, too.

28 There are some limitations on how PDFWriter handles embedded images. In particular, EPS is usually rendered at low resolution.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 284
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

systems, an alternative means of generating PDF is provided in the form of Acrobat Distiller, a program that takes PostScript as its input and generates equivalent PDF. Since almost any high quality layout program can produce PostScript, this mechanism is almost as general as using PDFWriter. If video, links, or sound are required, they must be added in a PDF editing program; Acrobat Exchange is provided for this purpose. Needless to say, PDFWriter and Acrobat Distiller and Exchange are not given away free.

PDF has had mixed fortunes. It has become a favoured format for distributing machine-readable manuals and other documentation, but it has not achieved the status of a universal portable format to which it aspires. Factors contributing to this lack of acceptance include the bulkiness of PDF files, relative to plain text or HTML, the unattractiveness of the interface to early versions of Acrobat Reader, and the limited multimedia support provided by Acrobat Exchange. Possibly, though, the need for PDF is not widely perceived: within a particular community, *de facto* interchange formats already exist; LaTeX within the academic technical community, MS Word in the business community, PostScript in printing and pre-press,[29] and so on. HTML, despite its shortcomings, perhaps already has some claim to be a universal portable document format, Web browsers being even more ubiquitous than Acrobat Reader. It has the further advantage that it can be generated with any text editor. For many purposes, now that CSS offers the potential for control over formatting, HTML is acceptable. It possesses the advantages of structural markup which we have previously extolled, and is easily distributable, being the native language of the World Wide Web.

29 Somewhat ironically, it is in pre-press that PDF has been most enthusiastically received, being easier to handle than PostScript.

Further Information

[Wil93] describes the principles of traditional book design; [Sas93], [Got98] and [Car97] consider, from different viewpoints, the impact of digital technology on those principles. [GQ99] includes an annotated version of the XML specification [XML]. [TDT96] is a short introduction to SGML from the writer's point of view. [Gra96] and [Gra97] are good descriptions of HTML and CSS, respectively; [HTMa] and [CSS] are the corresponding standards. [Ado93] describes the format of PDF documents.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 285
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Exercises

1. What advantages would be gained from re-defining HTML using an XML DTD instead of an SGML DTD? What would be the disadvantages?

2. By default, `EM` elements are usually displayed in italics. As we stated in the previous chapter, for display on screen, it may be preferable to use boldface for emphasis. Write a CSS rule to specify this appearance for `EM` elements.

3. How would you specify that an entire HTML document should be displayed with one and a half times the normal leading?

4. Write a CSS rule to set level 1 headings as 'call outs', that is, the heading text should be typeset in the margin with its top on a level with the top of the first paragraph of the following text. (You will need to specify a wide margin.)

5. For pages that use CSS extensively, it would be polite to display some text that could only be seen by browsers that could not interpret stylesheets, explaining that the layout was being ignored by the browser. There is no simple way of discovering whether a user agent supports stylesheets. Devise a means of embedding such a message in a page, using only HTML and CSS.

6. Construct a CSS stylesheet in which all the block-level and inline HTML elements described in this chapter are distinguished purely by their colour. Choose a colour scheme that is visually appealing, and ensures that people with defective colour vision can still distinguish the different elements.

7. Construct a simple example of a document that has a different meaning depending on whether or not a user agent correctly interprets stylesheets.

8. Find a Web page that uses visual layout tags and attributes. (You will have to view the HTML source to discover this.) Take a copy of the page, remove the visual layout and replace it with a CSS stylesheet that produces an identical layout on a suitably equipped browser.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 286
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

9. Rewrite your CSS stylesheet from the previous exercise to change every aspect of the page's layout without altering the HTML tags.

10. Could PDF be used as an alternative to HTML for Web page layout?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 287
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Hypertext

9

In Chapters 7 and 8 we described the character of digital text, one of the most important components of multimedia. As well as the simple text described there, multimedia productions often make heavy use of *hyper*text. Hypertext is text augmented with *links* — pointers to other pieces of text, possibly elsewhere in the same document, or in another document, perhaps stored at a different location. A navigational metaphor is usually employed to describe how these links work: the place a link occurs is called its source, the place it points to is its destination. A user 'follows' a link from its source to its destination: when a representation of the source is selected, usually by clicking with a mouse, the part of the document containing the destination is displayed, as if the user had jumped from the source to the destination, much as one follows a cross-reference in an instruction manual. Hypertext thus allows us to store a collection of texts and browse among them.

Because the user's interaction with hypertext is more complex than that with text, we must consider more than just appearance when we look at the interface to hypertext. As well as the question of how a link is to be displayed, we must consider whether and how to store a record of links that have been followed, so as to permit backtracking, and we must consider what additional navigational facilities should be provided. For most people, the most familiar example of hypertext is the World Wide Web, and the navigational facilities provided by the popular Web browsers illustrate the sort

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 288
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

of facilities that are required if users are to be able to browse a collection of hypertext documents comfortably.

You will almost certainly be aware, though, that the World Wide Web is not just made up of hypertext. Web pages can have images, sounds, video, and animation embedded in them, and scripts associated with them to perform interaction with the user. This network of media elements connected by links is an example of *hypermedia*, of which hypertext is a special case, restricted to the single medium of text. In describing hypertext in this chapter we will be laying the groundwork for a description of hypermedia in Chapter 13. Despite the added technical complexity of dealing with a multitude of media types, the generalization from hypertext to hypermedia is conceptually a simple one, so we can safely begin our technical discussion by considering hypertext. The development of the two is inseparable, so, as we wish to begin by reviewing this development, we will start by considering hypertext and hypermedia together.

A Short History of Hypertext and Hypermedia

Although popular perceptions of hypermedia focus on the World Wide Web, the concept has a long history. Its origin is generally traced to an article[1] written in 1945 by Vannevar Bush, in which he described a machine for browsing and annotating a large collection of documents. The *Memex*, as Bush's device was known, included a mechanism for creating links between documents, allowing documents related to the one currently being read to be retrieved, in much the same way as Web pages related to the current one can be accessed by following a link. The Memex was a mechanical device, based on photosensors and microdots, and it is hardly surprising that it was never built — digital computer technology provides a much more suitable mechanism for linked data retrieval.

1
V. Bush, "As We May Think", *Atlantic Monthly*, July 1945.

Bush's contention was that association of ideas was fundamental to the way people think, and that document storage systems should work in a way that reflects these associations, hence the need for links. This interest in matching retrieval to a hypothesized way of thinking persisted in the earliest work on computer-based hypertext: one of the first working hypertext systems was developed under the aegis of an institution with the grandiose

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 289
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

title of the Augmented Human Intellect Research Center.[2] Another feature of early writing and research on hypertext is the assumption that links would be added dynamically by different users of the system, so that the linked structure would develop over time on the basis of interactions between a community of users. There remains a considerable overlap between hypermedia systems and computer-supported collaborative working.

Among several experimental systems developed in the late 1960s and early 1970s, Ted Nelson's Xanadu project deserves mention, because it was intended to be a global system, storing the entire world's literature. In this sense, it foreshadowed modern distributed hypermedia systems, unlike the smaller scale, but more practical, systems that were developed in the years that followed. It was in connection with Xanadu that the word 'hypertext' was devised.

A number of experimental systems were developed throughout the 1970s and early 1980s. As experience with hypertext grew, some of these systems began to be used in earnest, particularly in information kiosks and similar applications. At this stage, much attention was paid to human interface issues, and to developing models of hypertext systems. Further work on these lines was rendered largely irrelevant by the arrival of *Hypercard* in 1987.

Hypercard, developed by Apple, was for several years distributed free with every Macintosh computer. The consequent sudden mass distribution of a hypertext system allowed Hypercard to establish itself as the paradigm of such a system — for a while, Hypercard 'stacks' become synonymous with hypertext — irrespective of any flaws in its interface and model of hypertext. Most, if not all, of the features to be found in Hypercard were derived from earlier systems, particularly Xerox's NoteCards; its importance lies not in innovation, but in popularization. Among those features were a card-based metaphor for organizing linked material, support for a variety of media, and the provision of a scripting language that allows actions to be associated with events and controls on a card. This language, HyperTalk, is sufficiently powerful to have allowed users to develop sophisticated applications in HyperCard — *Myst* is a HyperCard stack, for example.[3]

HyperCard brought hypermedia out of the laboratories and into the world; the World Wide Web made it into a cultural phenomenon. Unlike earlier systems, which used proprietary formats and were largely closed and self-sufficient, the Web uses publicly available

2 We leave it to the reader to judge whether the majority of contemporary Web sites can be said to augment the human intellect.

3 The last version of HyperCard was marketed as an application development tool, not a hypertext system.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 290
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

technology — anyone can create a Web page with a text editor, and plug-ins and helper applications allow browsers to handle arbitrary media types. This technology is, in fact, fairly crude, and it took over five years for the World Wide Web to evolve from a simple distributed hypertext system to a full hypermedia system with interactivity. The ubiquity of the World Wide Web means that its continuing evolution is likely to define the nature of hypermedia for some time to come.

The Nature of Hypertext

Text becomes hypertext with the addition of links which connect separate locations within a collection of hypertext documents. Links are active: using some simple gesture, usually a mouse click, a user can follow a link to read the hypertext it points to. To make this happen, a piece of software, called a *browser* is required.[4] Usually, when you follow a link, the browser remembers where you came from, so that you can backtrack if you need to. The World Wide Web is an example of a (distributed) hypertext[5] system, and a Web browser is a particular sort of browser.

4
A browser may be an independent program that reads hypertext documents as data, or it may be integrated with the document itself, to produce 'standalone' hypertext.

Browsers tend to encourage people to read hypertext in a *non-linear* fashion: instead of starting at the beginning and reading steadily, you might break off to follow a link, which in turn might lead you to another link that you can follow, and so on. At some point, you might go back to resume reading where you left off originally, or you may find it more fruitful to go on pursuing links.

5
Actually, hypermedia, as we have seen.

It is a common misconception that such non-linear browsing is the distinctive innovation of hypertext. The following quotation is an extreme expression of this view:

> "All traditional text, whether in printed form or in computer files, is sequential. This means that there is a single linear sequence defining the order in which the text is to be read. First, you read page one. Then you read page two. Then you read page three. And you don't have to be much of a mathematician to generalize the formula to determine what page to read next."[6]

6
Jakob Nielsen, from 'Usability considerations in introducing hypertext' in *Hypermedia/Hypertext*, ed. Heather Brown, Chapman and Hall (1991).

Leaving aside the mathematical invalidity of the proposed generalization, a glance at any magazine will show just how mistaken

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 291
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

is the notion of a single linear sequence for reading printed text. What we find is a collection of articles that can be read in any order, broken up by pages of advertisements, to be skipped over or glanced at between paragraphs; a contents page contains pointers to each article. Many magazines feature 'sidebars', with information tangential to the main text of the article; picture captions interrupt the linear flow of text; and articles may be split ('continued on page 52'), with their continuation many pages further on and parts of several other articles intervening.

Other forms of printed matter also display non-linearity. An encyclopedia is made up of short articles, which are often cross-referenced, so that having read one you may be led on to another. Cross-referencing is also common in manuals for software and in all sorts of technical reference material. The tiresome scholarly apparatus of footnotes and bibliographic references comprises a sort of system of links, and much academic research consists in following those links. Even novels, which usually have a linear structure are often read non-linearly: a reader might find it necessary to go back to an earlier chapter ('Was Piotr Ivan's brother-in-law or his faithful servant?') or even to go to the end to find out who murdered the Colonel; it may be necessary to stop reading the novel and resort to a dictionary to find out what 'tergiversate' means. Some novels explicitly use non-linear structures: Vladimir Nabokov's *Pale Fire*, and Flann O'Brien's *The Third Policeman*, for example, both include extensive notes, arranged in the same way as academic endnotes and footnotes, which must, therefore, be read out of the linear sequence of the text, but nevertheless play a major role in telling the story.[7]

It would not be going too far to say that, taken over all the varieties of reading, non-linearity is the norm when it comes to reading text. What is novel and distinctive about computer-based hypertext is the immediacy with which links can be followed, which creates a qualitatively different experience. Contrast waiting six weeks for a copy of a conference paper you have ordered on inter-library loan because you saw a reference to it in a journal with clicking on a link in a Web page and being shown the text of the same paper, as stored on a computer somewhere on the other side of the world, within a few seconds (on a day when Internet traffic is flowing briskly). Although text on paper does not have a unique linear sequence for reading, it does have a unique linear physical sequence, so following cross-references, even within a

7 It might be argued that the notes in *The Third Policeman* actually tell part of the story of the same author's *The Dalkey Archive*, thereby providing an example of inter-document linkage.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 292
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

single text, requires the reader to flip forwards and backwards between pages, using page numbers to identify the destinations of cross-references. Even though computer memory is also arranged linearly and addressed sequentially, layers of abstraction imposed by software on top of this arrangement, together with the speed of operation of modern computers and networks, makes this linearity transparent, so that hypertext systems can provide the illusion that links connect together disparate pieces of text into a network (you might even call it a web) that you can travel on.

Links

Hypertext raises some new issues of storage and display. How are links to be embedded in a document? How are their destinations to be identified? How are they to be distinguished typographically from the surrounding text? To appreciate the answers to those questions, we need a clear understanding of what a link is, and what sort of thing links connect.

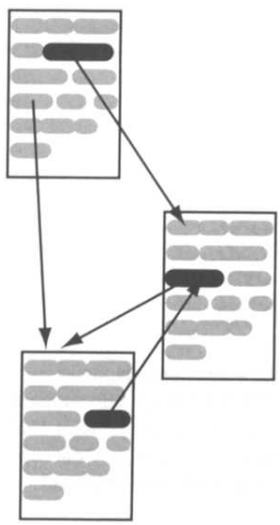


Figure 9.1
Simple uni-directional links

The simplest case is exemplified by the World Wide Web. If we confine ourselves for the moment to pages consisting purely of text, these are self-contained passages, which may be of any length, but usually fit on a few screens. Their only connection with other pages is through links (even though a link may be labelled 'next' or 'previous', such sequential connections between pages must be explicit). Within a page, though, the normal sequential structure of text is exhibited: you can sensibly read a page from beginning to end (although you don't have to), and elements such as headings and paragraphs are used to structure the content of the page. The HTML source of a Web page is often held in a file, but some pages are generated dynamically, so you cannot always identify pages with files, or indeed, with any persistent stored representation.

Hypertext systems are generally constructed out of self-contained elements, analogous to Web pages, that hold textual content. In general, these elements are called *nodes*. Some systems impose restrictions on their size and format — many early hypertext systems were built on the analogy of 3 by 5 index cards, for example — whereas others allow arbitrarily large or complex nodes.

In general, hypertext links are connections between nodes, but since a node has content and structure, links need not simply associate

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 293
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

two entire nodes — usually the source of a link is embedded somewhere within the node's content. To return to the World Wide Web, when a page is displayed, the presence of a link is indicated by highlighted text somewhere on the page, and not, for example, by a pop-up menu of links from that page. Furthermore, a link may point either to another page, or to a different point on the same page, or to a specific point on another page. Hence, Web links should be considered as relating specific locations within pages, and, generally, links connect parts of nodes (see Figure 9.1).

In HTML, each link connects a single point in one page with a point (often implicitly the start) in another, and can be followed from its *source* in the first page to its *destination* in the other. We call links of this type *simple uni-directional links*. XML and other more elaborate hypertext systems provide a more general notion of linking, allowing the ends of a link to be regions within a page (*regional links*), links that can be followed in either direction (*bi-directional links*), and links that have more than just two ends (*multi-links*). Figure 9.2 illustrates these generalized forms of link.

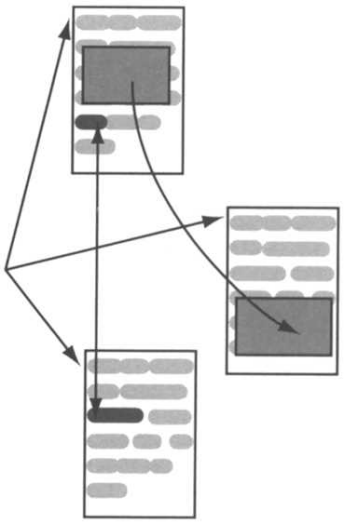


Figure 9.2
Regional, bi-directional and multi-links

Links in PDF

Adobe's Portable Document Format (PDF) supports hypertext linkage. PDF links are uni-directional, but not quite simple, since a restricted form of regional link is provided: each end of a link is a rectangular area on a single page. Since Acrobat Distiller and PDFWriter make it possible to convert just about any text document to PDF, and links can then be added using Acrobat Exchange, this means that hypertext links can be added to any document, however it was originally prepared.

➲ In PDF, links are considered to be annotations, that is, they are not part of the page's contents proper. This is reflected in the way links are implemented. A PDF document comprises a hierarchically organized collection of objects. (In a PDF file, the objects are stored in a textual form, using keywords to identify their type and attributes.) A document object consists of an `outline` object and a `pages` object, which is organized as a tree of `page` objects. Within each page, separate objects are used to hold the imageable content and the annotations. Link objects, containing the coordinates, in a device-independent coordinate space, of the source rectangle (on the parent page), a specification of the destination, and some information describing how links appear when the document

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 294
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

is displayed, are stored within the `annotations` object. If the destination is within the same document as the source, it may be specified in the `link` object as a page sequence number and the destination rectangle, together with additional information specifying the size at which the destination is to be displayed after the link is followed. If it is in another file, a separate `destination` object is used, which holds this information together with the name of the file containing the destination. A pointer to this `destination` object is then stored in the link. (This indirect form may also be used for links within a document; the special treatment of these internal links is provided for efficiency reasons, and to maintain compatibility with PDF version 1.0, which only supported links within a single document, and only allowed the destination to be specified directly within the link object.)

Links may be displayed by Acrobat Reader in a variety of styles. The default representation consists of a rectangle outlining the region that is the source of the link. When a user clicks within this rectangle it is highlighted — by default, colours within the region are inverted — and then the viewer displays the page containing the destination region. The actual destination, always a rectangular region, is highlighted. When the link is created, the magnification to be used when the destination is displayed can be specified, which makes it possible to zoom in to the destination region as the link is followed.

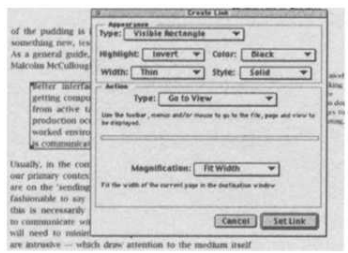


Figure 9.3
Adding a link to a PDF document

Adding links in Acrobat Exchange is a two-stage process. First, you select the link tool and use it to drag out the source region of your link. By default, this region is an arbitrary rectangle; by holding down a modifier key, you can constrain it to be the tightest possible rectangle that can enclose text that you select with the tool. Thus, sources can appear to be words in the text, if that seems appropriate. Once the source region has been selected, a dialogue box appears in which you can select the display styles for the source (see Figure 9.3[8]). In the second stage, you use the navigational controls of the program to move to the page containing the link's desired destination; if necessary, you can open a different file at this stage. If your destination is a specific region on this page, you can use a selection tool to draw its outline; otherwise, the top of the page is used as the destination. Once the destination has been selected, clicking on the button labelled `Set Link` causes the link to be stored.

8
The pop-up menu labelled Type in the `Action` pane will be explained in Chapter 14

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 295
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

Links in HTML

The links that can be embedded in a Web page composed in HTML are simple and uni-directional. What distinguishes them from links in earlier hypertext systems is the use of *Uniform Resource Locators (URLs)* to identify destinations.

➲ A URL uniformly locates a resource, but the concept of 'resource' thus located is quite hard to pin down.[9] In practice, a resource is anything that can be accessed by one of the higher level Internet protocols such as HTTP, FTP, or SMTP. Often, but by no means always, a resource is a file, or some data, but the way in which you can access that data is constrained by the protocol you use. For example, a `mailto` resource identifies a user's mailbox, but only allows you to access the mailbox by sending a message to it. An `ftp` resource identifies a file, and might even be used to identify a user's mailbox on systems where mailboxes are stored as files, but an `ftp` resource can be fetched over the network from its remote location. A resource is thus something like an abstract data type, identifying some data and providing a set of operations that can be performed on it.

9 The specifications are not unambiguously helpful here: "A resource can be anything that has identity [...] The resource is the conceptual mapping to an entity or set of entities, not necessarily the entity which corresponds to that mapping at any particular instance in time." (IETF RFC 2396 *Uniform Resource Identifiers (URI): Generic Syntax*). " [R]esource: A network data object or service that can be identified by a URL,..." (IETF RFC 2068 *Hypertext Transfer Protocol - HTTP/1.1.*

⇨ The HTML 4.0 specification stipulates the use of Uniform Resource *Identifiers* (URIs), rather than URLs. URIs are a superset of URLs, which also include *Uniform Resource Names* (URNs). URNs are intended to provide a location-independent way of referring to a network resource, unlike URLs which pin a resource down to a specific location for all time. In practice, Web pages invariably use URLs, so we will stick to the more familiar term and concept.

The URL syntax provides a general mechanism for specifying the information required to access a resource over a network. For Web pages, three pieces of information are required: the *protocol* to use when transferring the data, which is always HTTP, a *domain name* identifying a network host running a server using that protocol, and a *path* describing the whereabouts on the host of the page or a script that can be run to generate it dynamically. The basic syntax will be familiar: every Web page URL begins with the prefix `http://`, identifying the HTTP protocol. Next comes the domain name, a sequence of sub-names separated by dots,[10] for example `www.wiley.co.uk`, which usually identifies a machine within an organization, within a sector within a country. (In the US, the country is usually implicit, and the top-level domain is the sector — commercial, educational, government, etc.) There may be more

10 Occasionally, the numerical IP address is used here (see Chapter 15).

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 296
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

intermediate domains in a domain name than we have shown in this example: universities, for example, typically assign sub-domains to departments, as in `www.cs.ucl.ac.uk`, a Web server in the computer science (CS) department of University College London, an academic institution in the United Kingdom.

After the domain name in a URL comes the path, giving the location of the page on the host identified by the preceding domain name. A path looks very much like a Unix pathname: it consists of a /, followed by an arbitrary number of segments separated by / characters. These segments identify components within some hierarchical naming scheme. In practice, they will usually be the names of directories in a hierarchical directory tree, but this does not mean that the path part of a URL is the same as the pathname of a file on the host — not even after the minor cosmetic transformations necessary for operating systems that use a character other than / to separate pathname components. For security and other reasons, URL paths are usually resolved relative to some directory other than the root of the entire directory tree. However, there is no reason for the path to refer to directories at all; it might be some access path within a document database, for example.

⇨ In advertisements and other published material, you will frequently see the URLs of Web pages given with the leading `http://` omitted, and some Web browsers allow you to omit the protocol when typing URLs, making intelligent guesses on the basis of the rest. Although such usage in these contexts is sanctioned in the relevant standards, within an HTML document, you must always use a complete URL, or a partial URL of the form, and with the meaning, described below.

The only characters that can be used in a URL belong to the ASCII character set, because it is important that URL data can be transmitted safely over networks, and only ASCII characters can be considered safe for this purpose — and not even all ASCII characters. Certain characters that may get corrupted, removed or misinterpreted must be represented by escape sequences, consisting of a % character followed by the two hexadecimal digits of the character's ASCII code.[11] In particular, spaces must be written in a URL as %20.

11 Compare this with the QP encoding described in Chapter 7.

A URL as described so far identifies a Web page in one of three ways. In all cases, the domain name identifies the host running an HTTP server. The path might be a complete specification of the location of a file containing HTML, as in `http://www.wiley.co.uk/`

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 297
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

`compbooks/chapman/index.html` or, if it ends in a / character, the specification of the location of a directory. As a special case, if the path consists only of a /, it may be omitted: `http://www.wiley.co.uk/` and `http://www.wiley.co.uk` both identify the root directory of the UK mirror of John Wiley and Sons' Web site. Where a path identifies a directory, the configuration of the Web server determines what the URL specifies. Often, it is a file within that directory with a standard name, such as `index.html`.

The third way in which a URL identifies a Web page is via a program that generates the content dynamically. Again, the precise mechanism depends on the server's configuration, but usually such programs must reside in a specific directory, often called `cgi-bin`, or must have a special extension, such as `.acgi`. The occurrence of the letters CGI in both of these conventions indicates that a mechanism known as the *Common Gateway Interface* is being invoked. This mechanism provides a means for a server to pass information to and receive it from other programs, known as CGI scripts. CGI scripts are commonly used to provide an interface — or *gateway* — between the Web server and databases or other facilities. Several mechanisms are provided to pass parameters to CGI scripts, including appending a *query string* to the end of the URL used to invoke the script. The query string is separated from the path by a ?, as in `http://ink.yahoo.co.uk/bin/query_uk?p=macavon`.[12]

12 For more information on the Common Gateway Interface and CGI scripting see Chapter 15.

⇨ Although the Common Gateway Interface was intended to provide a standard for interaction between a Web server and other resources, many Web server programs and database systems provide their own methods for performing the same function. These proprietary solutions are often more efficient than CGI scripts, and have been widely adopted in commercial Web sites.

We have described a hypertext network as consisting of nodes connected by links, but the nodes (i.e. pages) of the World Wide Web are grouped into Web *sites*, each comprising a relatively small number of pages, usually held on the same machine, maintained by the same organization or individual, and dealing with related topics. Within a site, links pointing to other pages on the same site are common. *Partial URLs* provide a convenient shorthand for such local links.

Informally, a partial URL is a URL with some of its leading components (the protocol, domain name, or initial segments of the path) missing. When a partial URL is used to retrieve a

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 298
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

resource, the missing components are filled in from the *base URL* of the document in which the partial URL occurs. This base URL is usually the URL that was used to retrieve the document, although sometimes it might be specified explicitly (see below). The way in which base and partial URLs are combined is very similar to the way in which a current directory and a relative path are combined in hierarchical file systems, particularly Unix. For example, if the relative URL `video/index.html` occurs within the document retrieved via the URL `http://www.wiley.co.uk/compbooks/chapman/index.html` it will be equivalent to the full URL `http://www.wiley.co.uk/compbooks/chapman/video/index.html` Within the same document, `/catalog/` is equivalent to `http://www.wiley.co.uk/catalog/`. The special segments `.` and `..` are used to denote the current 'directory' (component in the hierarchy described by the URL path) and its immediate parent, so, again within the same document, the relative URL `../graham/index.html` is the same as `http://www.wiley.co.uk/compbooks/graham/index.html`. Generally, if a relative URL begins with a /, the corresponding complete URL is constructed by removing the pathname from the base URL and replacing it with the relative URL. Otherwise, just the last segment of the pathname is replaced, with `.` and `..` being interpreted as described.[13]

13
It is possible to describe this interpretation as a purely lexical transformation on the constructed URL. We leave this as an exercise.

⇨ A little thought will show you that, if relative URLs are resolved using the URL of the document in which they occur as the base URL, it is possible to move an entire hierarchy of Web pages to a new location without invalidating any of its internal links. This is usually something you want to be able to do, but occasionally you may need the base URL to be independent of the location of the document containing relative URLs. Suppose, for example, that you have constructed a table of contents for some Web site. You will want to use relative URLs, because all your links point to pages on the same server. If, however, you wish to be able to duplicate the table of contents (but not the site itself) on several sites, you will not want to use each copy's own location as the base URL for these relative URLs; instead, you will always want to use a URL that points to the site you are indexing. To achieve this, you can explicitly set the base URL in a document using the HTML `BASE` element. This is an empty element, with one attribute `href`, whose value is a URL to be used as a base for resolving relative URLs inside the document. The `BASE` element can only appear within a document's `HEAD`. Base URLs specified in this way take precedence over the document's own URL. Hence, if a document included the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 299
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046

element `<BASE href="http://www.wiley.co.uk/compbooks">`, then, no matter where the document itself was stored, the relative URL `chapman/index.html` would be resolved as `http://www.wiley.co.uk/compbooks/chapman/index.html`.

One more refinement of URLs is needed before they can be used to implement simple unidirectional links as we have described them. We emphasized that links connect parts of nodes, but a URL specifies a complete page. To identify a location within a page — a particular heading, or the beginning of a specific paragraph, for example — the URL needs to be extended with a *fragment identifier*, consisting of a # character followed by a sequence of ASCII characters. The fragment identifier functions roughly like a label in assembly language programs: it can be attached to a particular location within a document (we will see how shortly) and then used as the destination for a 'jump'. A fragment identifier is not really part of a URL: it is not used by HTTP requests, but is stripped off by the user agent making the request, which retains it. When the requested document is returned, the user agent will find the location designated by the fragment identifier, and display the document with that part of it in the window.

We have glossed over some of the fine detail of HTTP URLs — if you need to know everything about URLs consult the specifications — but the description just given should suffice to show that URLs provide the information required to access a Web page from anywhere on the Internet. The HTTP protocol provides a means of retrieving a Web page, given its URL, and therefore the combination of URLs and HTTP can be used to implement hypertext links in the World Wide Web provided we have a means of embedding URLs as links in HTML documents. This is provided by the `A` (anchor) element, which can be used as both the source and destination of a link, depending on which of the attributes `href`, `name` and `id` are provided.

The `href` attribute is used when an anchor is to serve as the source of a link. Its value is a URL, which may be absolute, if the link points to a document elsewhere on the World Wide Web, or relative, if it points to a document within the same hierarchy. The URL may have a fragment identifier appended to designate a specific location within the destination document, or it may consist solely of a fragment identifier, for links within the same document. The anchor element has content, which is displayed by user agents in some special way to indicate that it is serving as the source of a

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 300
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00046