hypertext link. For example, popular browsers by default show the content of anchor elements in blue and underlined; when a link's destination has been visited, a different colour (often green) is used to display the visited link whenever the document containing the source is displayed again within a designated period of time.

⇨ A CSS stylesheet can provide rules for formatting A elements, just like any other element. To cope with the desire to change the appearance of a link when it is visited, CSS provides special *pseudo-classes* `link`, `visited` and `active` that can be applied to A selectors. They identify subsets of the class of anchor elements, just as a class name does, but whether an anchor belongs to the subset depends on its condition in the browser, not on any attribute values. To distinguish pseudo-classes from ordinary classes, they are separated from the element name in selectors by a colon instead of a dot.

The following rules stipulate that links should be set in boldface until they have been visited, when they revert to the normal text font. While the user clicks on the link, it will appear extra large.

```
A:link {
    font: bold;
}
A:visited {
    font: normal;
}
A:active {
   font: xx-large;
}
```

Pseudo-classes, although they are only used for links in CSS2 provide a more general mechanism that can potentially be applied to other HTML elements.

Despite the elegance of this way of controlling the appearance of links, it should be used judiciously, if at all. Users rely on their chosen browser's default for the appearance of links in order to recognize them. Interfering with this recognition can detract from the usability of your Web pages.

An anchor element with a `name` attribute is presently the most common way of attaching a name to a location within a document. The value of the name can be any string that would be legal HTML, although when the name is used as a fragment identifier any characters other than the subset of ASCII that is legal in a URL must be escaped. Although an anchor attribute that is being

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 301
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

```
<H1>Links and URLs</H1>

<H2 id="links">Links</H2>
Hypertext links are implemented in HTML using the
&lt;A&gt; element and <A href="#urls">URLs</A>.
<P>
[etc, etc.]
</P>
<A name="urls"><H2>URLs</H2></A>
An introduction to the use of URLs in HTML can be found in
<A href=
 "http://www.w3.org/TR/REC-html40/intro/intro.html#h-2.1.1">
the HTML4.0 specification</A>.
They are the basis of <A href="#links">links</A>
in Web pages.
<P>
[etc, etc.]
</P>
```

Figure 9.4
Links and Anchors

used as the destination for hypertext links in this way has content, this is not usually displayed in any special way by user agents. An alternative to anchors with names is provided in HTML 4.0: *any* element may have an `id` (identifier) attribute; this may be used for several purposes, among them the specification of link destinations, by using the identifier as a fragment identifier. The use of identifiers instead of anchor names for this purpose has the advantage that any document element — for example, a header — can be used as the destination of a link, without the necessity of enclosing it in a dummy anchor. The disadvantage is that older browsers do not recognize identifiers.

There is nothing to stop you using the same anchor as both the source of a link, and a potential destination of other links, by providing it with an `href` attribute and a name or identifier. Within a document, each name or identifier must be unique, for obvious reasons. Although it would appear to be possible to distinguish between identifiers and names with the same value, this is not done, and it is not permitted for a name and an identifier to be the same.[14] The behaviour of a user agent in the face of name clashes is not defined.

Figure 9.4 shows examples of links within a document, using both named anchors and a header with an identifier; a link to a section of a separate document is also shown.

14 In computer language jargon, names and identifiers share the same namespace.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 302
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

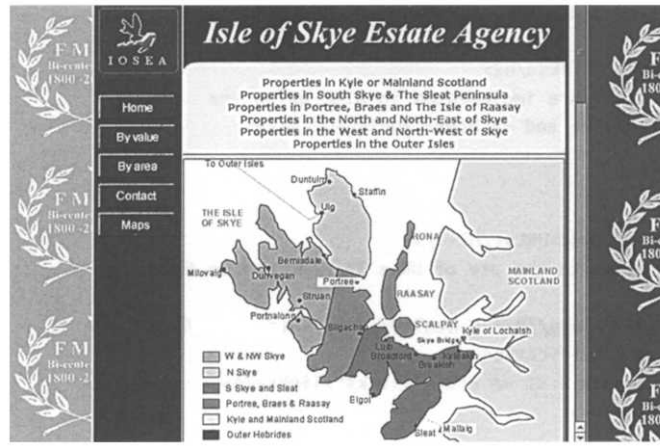


Figure 9.5
Frame-based navigation

Colloquially, we say that when a user clicks on a highlighted anchor element in a Web page, the browser 'goes to' the page that is the destination of the link; by extension, we talk about 'visiting' Web sites and pages. This metaphor of visiting has achieved almost universal currency, despite the fact that, in reality, the opposite takes place: we do not visit Web pages, they come to us. Clicking on a link that references a separate document causes the resource identified by the URL that is the value of the `href` attribute of the anchor to be retrieved via HTTP. Assuming that the resource is a Web page[15] the browser interprets the HTML and any associated stylesheet and displays the page. If a fragment identifier is appended to the URL, the browser will find the corresponding anchor (assuming it is there) and scroll the display to it. Usually, the newly retrieved page replaces any page being currently displayed by the browser, hence the idea of 'going to' the new page. Sometimes, though, the browser may open a new window.

15
We describe in Chapter 13 what may happen if it is not.

An interesting question arises if the link occurs within a frame being displayed as part of a frameset. Should the new page replace just that frame, or the entire frameset, or should it replace some other frame in the frameset? Any of these may be appropriate. Consider, for example, the typical frame layout shown in Figure 9.5, with a navigational menu in a small frame on the left, and the remainder of the window occupied by a page devoted to one of the topics on the menu. The menu will consist of links to the specific pages; since the intention of this style of layout is to ease navigation through the site by having the menu available all or most of the time, it is obvious that clicking on a link in the menu frame should cause

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 303
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

the retrieved page to be displayed in the main frame. If a page displayed in the main frame contains a link to another part of the same site, though, when it is followed, the new page should replace the old one. Finally, if a page contains a link to some other site, the navigational menu will become irrelevant when the link is followed, so the new page should be displayed in the whole window.

HTML provides control over the placement of retrieved pages within a frameset by allowing you to name frames, using the `name` attribute of the `FRAME` element, and specify the frame in which to display the destination of a link, using the `target` attribute of the `A` element. The value of this attribute may be the name of a frame, in which case, when the link is followed, the destination page is displayed in the corresponding frame. Alternatively, the target may be one of four special names, `_self` (the retrieved page is displayed in the frame containing the link), `_blank` (the retrieved page is displayed in a new window), `_top` (the retrieved page is displayed in the entire window, removing any frameset structure), or `_parent` (the retrieved page is displayed in the frameset containing the frame containing the link; this is the different from `_top` in the case of nested framesets). If no target is specified, `_self` is assumed. In a frameset layout such as that of Figure 9.5, the main frame would have a name such as `mainframe`, which would be used as the target for links in the navigation menu. Links within the site specify no target, so their destinations are displayed in the same frame as their sources, typically the main frame. Links which point beyond the site would have a target of `_top`, so that this frame structure would be effaced when a user leaves the site.

⇨ HTML anchors provide a flexible and general linking mechanism, but this very generality may be its weakness. There is nothing to suggest why a link has been established; in other words, simple `A` elements do not capture any of the semantics of linking. Thus, a user can have no way of knowing what sort of relationship the link expresses, and hence whether it is worth following, and programs cannot provide automatic navigational hints based on, for example, similarities between a link and those previously followed. The `rel` (relationship) and `rev` (reverse relationship) attributes are an attempt to capture more of links' semantics. Their value is a *link type*, which is a string designating a type of relationship that may be expressed through linkage. For example, an occurrence of a technical word in a document may be linked to an entry in a glossary; the `rel` attribute of the source anchor for this link could have the value `glossary`,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 304
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

as could the rev attribute of the destination anchor at the entry for that word in the glossary:

```
... by means of a
  <A href="../terms.html#url" rel="glossary">URL
</A> ...
```

and

```
<DT><A name="url" rev="glossary">URL</A>
  <DD>Uniform Resource Locator ...
```

A related issue concerns relationships between entire documents, such as one document being a translation of another into a different language, that are not expressed naturally by hypertext links between anchors within documents. The LINK element is provided to express such relationships. LINKs can only appear in a document's HEAD; they have an href attribute, whose value is a URL identifying the linked document, and may have rel or rev attributes, serving the same function as they do with anchors. Since they are in the document's head, link elements are not displayed by browsers — in fact, they have no content. It is envisaged that they might be used to construct special purpose menus or toolbars to assist with navigation among a set of related documents.

A collection of standard link types is defined to designate common relationships between documents, such as Next, Prev, and Start, which can be used to connect together a sequence of documents, or Chapter, Section, Subsection, and Contents which can be used to organise a collection of documents in the form of a conventional book. We have previously seen the use of a LINK element, with a rev value of Stylesheet to connect a document to an external stylesheet definition. In addition to the standard relationships, Web page authors are free to invent their own, although these are of less use in helping programs and users understand the structure of a collection of documents.

Unfortunately, capturing the semantics of linkage seems to have little appeal to Web authors and browser vendors: rev and rel have not been widely adopted, and with the exception of the special case of stylesheets, most browsers ignore <LINK> tags. The functionality of these HTML features has been incorporated into XML's linking facilities, described in the next section, where they will hopefully fare better.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 305
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

XPointer and XLink

XML allows you to define your own document elements, and following the precedent of HTML, it is natural to want to include linking elements among them. It would be simple to stipulate that attributes whose value is a URL (or URI) should be treated as hypertext links, but this would mean that XML was restricted to the same class of simple uni-directional links as HTML provides. This was felt to be inadequate, so a more general mechanism has been proposed. It consists of two parts: XPointer extends HTML's fragment identifier, and XLink generalizes its anchors. At the time of writing, these proposals have not been adopted as W^3C Recommendations, so the following description should be treated as an outline of a possible way of implementing the more general forms of link identified earlier in this chapter.

Adding a fragment identifier to a URL used as the value of the `href` attribute of an HTML anchor allows you to link to a specific named anchor or element with an identifier in a document. If there is no such element, you cannot create a link. XPointer defines a syntax that enables you to describe the destination of a link in terms of its position within the document. This means, among other things, that you can create links to arbitrary points in documents whose source you do not have access to. In addition, XPointer provides a way of defining a region within a document, so that it supports regional hypertext links.

As we pointed out in Chapter 8, every element of an HTML document has a unique *parent* element. The same applies to XML document elements. As a consequence, the structure of a document can be represented in the form of a tree, with each element being a child of its parent. Any element can be identified by its position in the tree, relative to the root or a named node, such as 'the second paragraph after the third level two header', or 'the third child of the second element after the element with id "orinoco" that has the same parent'. Xpointer is a means of expressing such positions in a way that can easily be processed by software. XPointer expressions — themselves known as XPointers — are appended to URLs in the same way as fragment identifiers are. (Recall that fragment identifiers are stripped off from a URL before the resource is retrieved, so altering the syntax of what may follow the URL proper does not interfere with protocols that interpret URLs.)

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 306
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

An XPointer is a sequence of terms, separated by dots. You can think of it as a set of instructions giving you directions to follow in the order specified to reach the intended point in the document. The first term usually refers to a fixed location, such as an element with an id, and subsequent terms tell you how to get from there to where you are going, by choosing, for example, the third child or the fourth paragraph. Every term consists of a keyword followed by some brackets, usually containing arguments. The terms `root()` and `id(`*name*`)` are used to establish the starting points for XPointers. The first form refers to the outermost element of a document — the HTML element of an HTML page, for example; the second refers to an element with an `id` attribute whose value is *name*,[16] any legitimate XML name. This latter form is similar to the use of fragment identifiers in HTML; for the specially common case of HTML anchor nodes with name attributes, a term of the form `html(`*name*`)` can be used. Thus, the XPointer `id(orinoco)` identifies the element (which should be unique) whose `id` attribute's value is `orinoco`, while `html(tobermoray)` identifies an element with start tag `<A name="tobermoray">`.

16
Since XML allows you to redefine almost everything, the situation is actually slightly more complicated, but most of the time this will be the case.

Once a starting point has been established, relative motion can be specified by terms using the keywords `child`, `descendant`, `ancestor`, `preceding`, `following`, `psibling`, or `fsibling`. The intent of most of these should be evident: if `E` is an element, a child is any element that is immediately contained in `E` (i.e., `E` is its parent), while a descendant is any element within the content of `E`; conversely, an ancestor is any element within whose content `E` appears. Elements are considered to precede `E` if their start tags appear before it, and to follow it if their end tags appear after it. The siblings of `E` are those elements with the same parent: `psibling` selects siblings that precede `E`, `fsibling` those that follow it. Figure 9.6 illustrates these relationships in terms of the tree structure of a document.

These keywords take between one and four arguments. The first is a number, allowing a particular element to be identified by counting. For example, the term `child(3)` identifies the third child of a node. The second argument, if it is present, is an element type name, and restricts the counting to elements of that type: `child(3, P)` identifies the third child that is a `P` element. The special value `#element` can be used to mean 'any element': `child(3, #element)` is thus the same as `child(3)`. The remaining two arguments are an attribute name and value, and restrict matters further to elements

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 307
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

with the specified attribute having the given value: `child(3, P, lang, de)` identifies the third paragraph child whose `lang` attribute has the value "de".

Although this description has been slightly simplified, it should suffice to show you that the examples cited at the beginning of this section can easily be expressed as XPointers: 'the second paragraph after the third level two header' is `root().child(3, H2).following(P,2)` (assuming HTML element names) and 'the third child of the second element after the element with id "orinoco" that has the same parent' is `id(orinoco).fsibling(2).child(3)`. (You have to read them backwards to get the English version, but navigation in the tree is specified from left to right.)

So far, XPointers have enabled us to specify single elements in a document. Another keyword, `span`, allows them to specify regions that span more than one element. It takes two XPointers as arguments, with the effect of identifying the region extending from the element identified by the first through that identified by the second. For example `id(orinoco).span(child(3), child(6))` identifies the third through sixth child elements of the element identified as "orinoco".

Finally, XPointers include a crude facility for identifying positions by text content: a term of the form `string(`*n, string*`)` identifies the n^{th} occurrence of the string *string*. For example, `root().string(3, "fish")` points to the third occurrence of the string "fish" in a document. (Additional embellishments, which will not be described here, allow you to refine string terms to point to, for example, the three characters following a string, and so on.)

XPointers extend the power of links by providing a flexible means of locating elements within documents. XLink extends their power in a different direction, by generalizing the notion of what a link is. It does this in a way that may be familiar from mathematics. A link is a connection or association between two locations. In the abstract, we can consider the ordered pair comprising the link's source and destination to *be* the link — nothing extra is really required. If we want bi-directional links, we only need to forget the ordering of the pair, so neither location is distinguished as either source or destination, but they remain connected. Once you have got this far, allowing as many locations as you want to be part of a link is a relatively small next step — in the abstract world. In practice, it raises some thorny implementation issues.

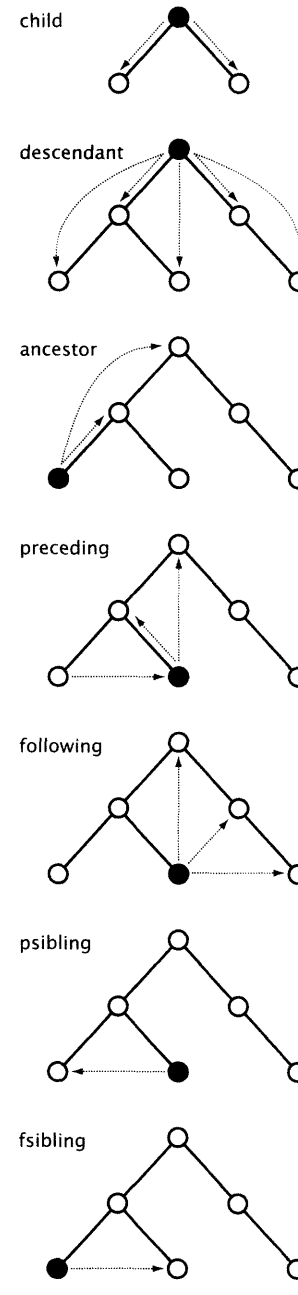


Figure 9.6
Relationships within a document's tree structure

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 308
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

XLink provides two sorts of link: *simple* and *extended*. Simple links are almost the same as HTML links and will not be described here. Extended links correspond to the generalization described in the previous paragraph. Syntactically, an extended link element needs to specify several locators (URLs or XPointers); since an element cannot have more than one attribute with the same name (such as `href`), XLink proposes separate element types to denote the presence of an extended link and the locations being linked, the latter appearing in the content of the former. If we assume that a DTD defines element types `exlink` and `locator` (leaving aside the question of how these are to be identified as link and locator elements), an extended link might look like this:[17]

17
In XML the tags for empty elements end with />, as in the `locator` elements here.

```
<exlink>
  <locator link = "../text.xml"/>
  <locator link = "../graphics.xml"/>
  <locator link = "../video.xml"/>
  <locator link = "../animation.xml"/>
  <locator link = "../sound.xml"/>
</exlink>
```

This is all very well — an extended link is a collection of linked resources, as required — but where should elements such as `exlink` appear? In one of the linked documents? In all of them? In a separate file? The XLink specification is rather vague on the matter. It provides for the possibility that a link might be `inline` or `out-of-line`. Technically, an inline link is one that includes itself as one of the linked objects; practically, this means that it appears in one of the documents being linked. An out-of-line link does not, so typically out-of-line links will be stored in separate linking files, devoted to the links connecting a group of documents. This begs the question of how out-of-line links are to be located by user agents.

For the common case where several documents can be considered as a group, with a collection of out-of-line links connecting parts of them in different ways, the links can be stored in an *extended link group* element, and a specialized locator called an *extended link document* element can be placed in each of the documents in the group to point to the links.

The idea of extended links is not new, but previously its implementation has been confined to hypertext systems on a smaller scale than the World Wide Web. It remains to be seen whether succesful implementations can be constructed for the Web, and whether extended links will be considered a worthwhile extension by Web users.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 309
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Navigation and Structure in Hypertext

The possibility of disorientation — being 'lost in hyperspace' as the cliché happily puts it — has been anticipated since the earliest experiments with hypertext. The division of text into nodes leads to fragmentation and a disruption of the context supplied by conventional linear structures; the multiple branching paths provided by links, while offering ample opportunity to browse endlessly, can resemble a maze to a user searching for one specific piece of information.

Much early thinking about this problem concentrated on the provision of graphical views[18] of the link structure between a collection of nodes. This approach has many problems — for any non-trivial collection of nodes and links the diagram itself becomes a confusing tangle. It is difficult to provide enough information about a node's content for users to determine whether it is interesting, or remind them whether they have visited it before — and, despite various ingenious embellishments, such as the use of 'fish-eye' views, showing nodes close to the current one in more detail than those further away, these graphical navigation aids have been largely abandoned in favour of navigation based on browsing history, indexing and searching.

18 Often confusingly called 'browsers' in the literature of the time.

Often, browsing in hypertext entails following side-tracks from a linear sequence and then returning to pursue it further. Providing a 'go back' command is a natural way of supporting this mode of reading. To do so requires the browser to maintain a stack of recently visited nodes, so it is straightforward to extend the facility by making this stack visible to the user in the form of a *history list*, and allowing users to jump directly to any node on the list, going back an arbitrary number of steps. A simple 'go back' facility provides a form of backtracking whereby a path is followed through the hypertext until it is no longer fruitful, and then the user backs up to a point where there was a different choice of link to follow and tries another path. In an alternative way of working, direct transfers to nodes on the history list allow a collection of recently visited nodes to be treated as a cluster, with arbitrary jumps being made between them. In effect, the history list provides an *ad hoc* set of dynamically constructed links.

History lists are always of limited size. Even if they were not, a complete history of your browsing activity will soon become

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 310
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

as confusing to find your way around as the hypertext network itself. It is more useful to summarize this activity over a period of time in the form of a collection of pointers to previously visited interesting nodes. On the World Wide Web, pointers are URLs, and, since these are just text strings, maintaining a collection of *bookmarks*, as URLs of interesting sites are usually called[19] is simple. Typically, Web browsers allow you to bookmark a page when you visit it, recording its URL together with the page title (this can be replaced by a mnemonic name subsequently, if the title is not sufficiently meaningful); bookmarks can be organized into a hierarchy and sorted. A menu of bookmarks is provided by the browser, which shows their names; selecting a bookmark from this menu causes the browser to retrieve the corresponding page. Instead of following trails of links, users can construct a collection of bookmarks that enables them to go directly to their chosen sites. This way of working is extremely common: you will notice that it entails rejecting the browsing approach to finding information in favour of direct access to a relatively samll number of familiar sites.

19
Internet Explorer prefers to call them 'favourites'.

Browsing is based on looking for information by its associations. An alternative, and often more natural procedure, is to look for information on the basis of its content. This approach is closer to information retrieval and database querying; it can serve as a way of finding a starting point for browsing as well as being an alternative to it. In a small hypertext network, stored on a single computer, it makes sense to search through the whole network for a keyword or phrase by applying a conventional text searching algorithm to each of the stored nodes. For larger networks, and especially for distributed networks such as the World Wide Web, this approach is not practical. Instead, we would like to construct an index (which can be kept on a single Web site), consisting of keywords together with URLs of nodes that they describe, and search that. Such an index could also be organized into categories using a suitable classification scheme, so that it can serve as a subject index, rather like a library's classified catalogue.

Many small indexes have been constructed simply by users publishing their bookmarks, sometimes supplying a simple CGI script to search through them. Where these form a coherent set of pages on a single topic, for example MPEG video, the result can be a useful access point for specialized browsing on that topic. Wider coverage is available from commercial index sites, such as Yahoo! and AltaVista. Here, huge numbers of URLs have been

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 311
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

collected, classified, and provided with a powerful search engine that supports the full range of Boolean query operations normally found in information retrieval systems.

There are two broad approaches to constructing index sites: manual and automatic. Using the manual approach, sites are classified on the basis of people's evaluation of their content. A URL for a Web site is normally added to the index in response to a request from its author, who suggests an initial position in the classification — usually a hierarchy of topics — and supplies a brief description of the contents. The site is then visited, and the classification is refined if necessary. The description is kept along with the URL, so it is possible to find a site either by navigating through the classification hierarchy, or by searching through the descriptions for keywords.

⇨ The search facility helps to compensate for the limitations of hierarchical classifications, which are well known to information scientists and will be familiar if you have ever used a hierarchical library catalogue: a site cannot be in more than one place in the hierarchy[20] even though there is often good reason to classify it in more than one way. For example, should a Web site devoted to the use of MPEG-2 video for broadcast television be classified under television, which may be a sub-category of entertainment, or under digital video, a sub-category of multimedia within the computing category? Evidently, a great deal depends on the structure of the hierarchy. Most classifications used by Web index sites are heavily slanted towards computing, communications and physical sciences, reflecting the origins of the World Wide Web. For example, Yahoo!'s top-level categories include one devoted entirely to computers and Internet, while another is devoted to everything concerning business and economy.[21] Such imbalances may be alleviated by replacing the simple hierarchical classification by one of the more sophisticated schemes developed by information scientists, or by changing the structure of the classification as the content of the material being indexed changes. Whereas libraries' classification schemes tend to ossify, because of their relationship to the physical books and their shelving, a Web index, which exists and deals with objects that exist only on computers, can be changed much more easily.

20 Although most indexes support the use of aliases that can make it appear almost as if it were.

21 And computer science is a sub-category of science.

While the manual approach to constructing indexes has the considerable advantage that human intelligence is applied to classifying and filtering the entries, it is labour-intensive — to be done properly, it requires somebody to actually go and visit each Web site submitted for indexing — which makes it difficult for the indexes to keep up with the rapid pace of change of the World

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 312
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Wide Web. Automatically constructed indexes avoid this problem. They are built by programs variously known as *robots*, *spiders* or *Web crawlers*, which simply follow links, collecting URLs and keywords from the pages they encounter. The keywords from a page are then used as the basis for its classification, and for searching. The success of this approach depends on the robot's success in extracting meaningful keywords, on the heuristics used to classify sites on the basis of keywords, and on the efficiency and sophistication of the search engine used to query the database of keywords and URLs.

AltaVista is the oldest and probably still the largest of the automatically constructed indexes. Actually, construction is not entirely automatic: for a Web site to be included, the URL of its root page must be explicitly submitted to Alta Vista. Subsequently, the Alta Vista spider[22] will retrieve that page and, by following links, all the other pages on the site. It then indexes all the text on those pages. A highly efficient search engine running on some very powerful computers is used to process queries against the database of indexed words. When a user submits a query containing any of the words found on a page, the first few lines of that page, together with its URL, are displayed among the query results.

22 Actually, they call it a Superspider.

This brute force approach to indexing can produce some idiosyncratic results, as you will probably know if you have ever used AltaVista. The spider cannot in any sense understand natural language, or differentiate between different uses of a word in different contexts, so, for example a search for 'the blues' will find Web pages devoted to Birmingham City Football Club and the St. Louis Blues NHL hockey team as well those dealing with blues music.[23] It is also possible, though, that the actual text of a page may not include the words one would normally use to describe its subject: that may be implicit in the content, or the author may have used a synonym. (For example, the paragraph preceding this one does not use the word 'robot', only 'spider'.) Furthermore, the opening lines of text from a Web page do not always provide a very helpful indication of its content. The current state of the art in natural language understanding by software does not provide a practical way of improving matters. Instead, the problem has to be approached from the other end by providing a means of adding a description that can be easily processed by a simple program to a Web page.

23 You also get an alarming number of sites devoted to the Moody Blues, but it wouldn't be fair to blame the search engines for that.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 313
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

HTML uses the META element for this purpose. META is a general mechanism for specifying *metadata* — data about data; in this case, data about a page, as against data that is part of the content of a page. A description for the benefit of indexing software is just one useful example of metadata. (PICS ratings are another.) META is an empty element that can only appear in the head of an HTML document. Two attributes, name and content, are used to allow it to specify arbitrary metadata as a pair of values, the name serving to identify the type of information, the content providing the corresponding value. This scheme avoids a proliferation of elements and leaves the maximum flexibility for choosing properties to use to describe a document.

➲ The inevitable drawback that accompanies this flexibility is that metadata lacks any structure. The proposed *Resource Description Format (RDF)* is intended to address this drawback, by incorporating ideas from conventional database modelling and object-oriented technology to provide an architecture for metadata that retains the flexibility of the simple META element approach, but allows metadata to be organized more formally.

AltaVista suggest the use of two named properties to assist their spider: description and keywords, whose intent is clear from the chosen names. The description, if present, is used instead of the first lines of the page when the results of a query are displayed. The keywords provide additional index terms, which may not be present in the text itself. Thus, if a Web page was devoted to the life and times of the blues singer Robert Johnson, it could usefully include the following:

```
<META name="description"
      content="The life and times of the
               blues singer Robert Johnson">
<META  name="keywords" content="blues, music,
                    King of the Delta Blues">
```

➲ Instead of the name attribute, a META element may include an hhtp-equiv attribute, which causes the corresponding content value to be incorporated into HTTP responses as a header. For example, PICS data can be incorporated into an HTML document using a META tag with its http-equiv attribute set to PICS-Label. This produces a Pics-Label header in an HTTP response for that page, with the value of the META element's content attribute as data. This way, metadata can be retrieved using an HTTP HEAD request, so the page itself does not have to be downloaded.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 314
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

⇨ Sites that provide an index to the World Wide Web are naturally popular, since they provide a starting point for browsing as well as direct access to specific information one may be looking for. By adding additional services, such as a news feed and on-line shopping, to the basic indexing functions, sites such as Yahoo! and AltaVista are evolving into *portals*: ways in to the tangle of the rest of the Web that, for some users, may remove the need for browsing entirely. As a result, they are among the few Web sites that are guaranteed to be visited many times, and hence they are attractive to advertisers. They are becoming seen as an answer to the question that has vexed business since the commercialization of the World Wide Web: how can you actually make money out of it?

If, as we suggested at the beginning of this section, disorientation in hypertext is due, at least partly, to the loss of contextual structure that results from the fragmentation of text into nodes, then one solution must lie in the development of new contextual structures appropriate to hypertext. In this connection, the concept of a *home* node has been used in many hypertext systems. For a Web site, it denotes the first page that is normally retrieved from that site; generally, it denotes a similar entry point to a collection of linked documents. Most nodes in the collection have a link back to the home node, so that it is always possible to return to a known point in the network.

Very often, a Web site or hypertext document collection has a hierarchical structure, usually with cross-links. The major divisions of the hierarchy provide an organizational and navigational framework. Within such an organization, the home node can be seen as the root of a tree of nodes; it will contain links to the roots of the sub-trees containing nodes belonging to the major sub-divisions of the site. These in turn may contain links to nodes at a lower level in the hierarchy. A frame-based display, such as we saw earlier in this chapter, can be used to provide navigation within a hierarchy: a navigational frame containing links to the roots of all the major sub-trees and the home node can be continuously displayed so that, at any time, a user can move between divisions. In a more sophisticated variation, the navigational frame can change so that, as well as the top-level links, it always contains links to important nodes within the sub-tree currently being explored. In the absence of frames, each node must explicitly contain links to the major divisions (again, a form of organization that should be familiar). This is less satisfactory, because of the duplication of link information and the consequent

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 315
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

problems of maintaining consistency when changes are made to the structure of the hypertext network.

Hierarchical structures do not always make sense, and sometimes a simple sequential structure is preferable. Here, every node needs a link to the previous and next one in the sequence, and possibly a link to the start of the sequence. Often a hybrid structure with sequential and hierarchical components will fit the material best.

On a scale larger than a single Web site or document collection, few structures have been developed. The index sites previously described serve, in some ways, as higher level home nodes, but it is not clear how navigational assistance, analogous to navigation frames, can be provided on this scale. Some *ad hoc* structures that make it easier to find one's way around a collection of Web sites have been developed, most notably, the 'Web ring', where a set of site maintainers agree to add links to their sites so that they form a circular sequence: once you have arrived at one of them, you will be led around the whole ring.

The structures just described possess little of the richness of long-established textual strategies used by writers to organize their material. The complexity and flexibility of language offers possibilities that an essentially crude system of nodes and links cannot approach. Despite the claims that have been made for hypertext's non-linearity freeing readers from the limitations of conventional text, in the end, hypertextual structures are far more restricted. The usefulness of hypertext (and hypermedia) lies in the way it allows us to add connections, not remove linear restrictions.

Further Information

Although it is dated, [Con87] remains a useful survey of hypertext issues and early systems. [BLFM98] contains the formal definition of a URL; the HTML references given in Chapter 8 describe how they are used. We will describe the mechanisms underlying World Wide Web links in Chapter 15. The specifications for XPointer and XLink can be traced from [XML].

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 316
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Exercises

1. Show how you can simulate bi-directional links and multi-links with simple uni-directional links. What advantages are there in using the more general forms of link instead of simulating them in this way?

2. In PDF, the source and destination of a link are arbitrary rectangular areas that may contain some text. Why cannot HTML use the same model for anchors?

3. Regional links allow for the possibility of links whose sources overlap. How would you cope with the problems this causes?

4. In a frame-based Web site, why might a target other than `_top` be used for links which point beyond the site?

5. According to Figure 9.6, a node's parent both precedes and follows it. Explain how this can be so.

6. Assuming HTML tags and attributes, write down XPointers for the following:

   (a) The third level 1 header in a document.

   (b) The second paragraph after the anchor with name `References`

   (c) The third anchor element in the fourth paragraph following the second level 2 header.

   (d) The first span element in the same paragraph as the anchor with name `References`.

7. Suggest a way of displaying the structure of a Web site that would provide assistance with navigation, assuming the site has (a) 10, (b) 100 and (c) 1000 pages.

8. Devise a classification scheme for Web pages that avoids the shortcomings of hierarchical schemes identified in this chapter.

9. Give examples showing how metadata could be used to assist search engines in avoiding (a) false negatives and (b) false positives.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 317
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

10. The *Encyclopedia Britannica* and several good dictionaries are now available online, so, in principle, every term used in any Web page could be linked to its definition in one of these reference works. Would this be (a) practical and (b) useful?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 318
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

10 Video



All current methods of displaying moving pictures, whether they are on film or video, broadcast to a television, or stored on a computer system or digital storage medium, depend on the phenomenon known as *persistence of vision* — a lag in the eye's response to visual stimuli which results in 'after-images' — for their effect. Because of persistence of vision, if a sequence of still images is presented to our eyes at a sufficiently high rate, above what is called the *fusion frequency*, we experience a continuous visual sensation rather than perceiving the individual images. If consecutive images only differ by a small amount, any changes from one to the next will be perceived as movement of elements within the images. The fusion frequency depends on the brightness of the image relative to the viewing environment, but is around 40 images per second. Below this frequency, a flickering effect will be perceived, which becomes more pronounced as the rate at which successive images are presented is decreased until, eventually, all illusion of motion is lost and we see the sequence of still images for what it really is.

⇨ Contemporary film is shot at a rate of 24 frames per second, but film projectors are equipped with a device that interrupts the projection, effectively displaying each frame twice, so that the projected frame rate is 48. In countries that use the NTSC system for television and video, frames are displayed at a rate of roughly 30 per second, but again the rate at which the picture is refreshed is effectively doubled, because each frame is broken into two interlaced halves,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 319
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

known as fields, which are displayed one after the other (see below). The PAL and SECAM systems work in a similar manner, at a rate of 50 fields (25 frames) per second. When we are displaying images on a computer monitor, we can choose the rate at which they are displayed. The relatively high refresh rate of the monitor avoids flicker, so digital display does not require interlaced fields, but images must still be displayed rapidly enough to produce the illusion of motion. A rate of 12–15 frames per second is just about sufficient, although higher rates are to be preferred.

There are two different ways of generating moving pictures in a digital form for inclusion in a multimedia production. We can use a video camera to capture a sequence of frames recording actual motion as it is occurring in the real world, or we can create each frame individually, either within the computer or by capturing single images one at a time. We use the word *video* for the first option, and *animation* for the second. Since both video and animation are forms of moving pictures they share some features, but there are also technical differences between them. For live-action video it is necessary both to record images fast enough to achieve a convincing representation of real-time motion, and to interface to equipment that conforms to requirements and standards which were defined for broadcast television — even though, when we come to play back the video on a computer, these requirements and standards are largely irrelevant. To make matters worse, there are several different standards in use in different parts of the world.

As we will see, video places considerable strains on the current processing, storage and data transmission capabilities of computer systems. Video is also much in demand from consumers, whose expectations are typically based on their experience of broadcast television; it is probably the area of multimedia which is presently most subject to technical change. Typically, video targetted at consumer equipment will be played back at reduced frame rates, possibly even with jittering from dropped frames, in windows much smaller than the smallest domestic television set, and it will often exhibit visible compression artefacts. In order to accommodate the limitations of low-end PCs, considerable compromises over quality must be made. Video should therefore be used judiciously in multimedia productions intended for low-end platforms. Consumers who are used to broadcast digital TV will not be impressed by what has been disparagingly, but not unjustifiably, described as 'dancing postage stamps'. Good video production for general consumption



Figure 10.1
A sequence of video frames

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 320
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

therefore requires both a careful choice of material and a mode of presentation that does not draw attention to its defects.

Digitizing Video

The one thing that you need to keep constantly in mind when considering digital video is its size. A video sequence consists of a number of frames, each of which is a single image produced by digitizing the time-varying signal generated by the sensors in a video camera. We invariably use bitmapped images for video frames (because that is what video equipment generates). The size of the image produced for each frame of NTSC video, as used in North America and Japan, is 640 pixels wide by 480 pixels high. If we use 24-bit colour, each frame occupies $640 \times 480 \times 3$ bytes, which is 900 kilobytes. One second of uncompressed NTSC video comprises almost exactly 30 frames, so each second occupies just over 26 megabytes, each minute about 1.6 gigabytes. The figures for the PAL system used in Western Europe and Australia are slightly higher: 768×576 at 25 frames per second gives 31 megabytes per second or 1.85 gigabytes for each minute. At these rates, you could not get very much video on to a CD-ROM or DVD; nor could you transmit it over any but the fastest network — certainly not the Internet as we know it. By using a good sized disk array, you would be able to store an entire feature film, and the required data transfer rate is within the reach of the latest SCSI standards. Storing video in such a form is therefore a feasible option for film and TV studios, but not (yet) for the domestic consumer.

Notice that the need for very high rates of data transfer arises from the fact that these are *moving* pictures, and we must deliver them one after another, fast enough for them to be perceived as a representation of continuous motion. The high volume of each frame arises from the storage requirements of bitmapped images. As we saw in Chapter 5, we can reduce this requirement by applying some form of compression to our images. It should come as no surprise, therefore, to learn that we also apply compression to video. For transmission over a relatively slow network such as the Internet, or for playback directly from CD-ROM, we must apply severe compression, as well as using other methods, such as limiting the frame size, to reduce the volume of data. This

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 321
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

results in a significant loss of quality, despite the claims that are sometimes made to the contrary. For capturing video in real-time, this compression must be carried out so fast that dedicated hardware is needed. In the remainder of this discussion, we will ignore the possibility of working with uncompressed video, since this is only feasible on very high-end equipment at the present time.

Digital video may either be captured directly from a camera, or indirectly from a video tape recorder (VTR) or (leaving aside questions of legality) from a broadcast signal. Current technology offers two sites for performing the digitization and compression: in the computer, or in the camera. If a VTR intervenes, or a signal is broadcast, the nature of the signal is not changed by the recording or broadcasting process — digital VTRs record digital signals, analogue VTRs record analogue signals.

In the case of digitization being performed in the computer, an analogue video signal, conforming to some broadcast video standard, is fed to the input of a *video capture card* attached to the computer. Within this card, the analogue signal is converted to digital form. Usually, the resulting digital data is compressed in the card, before being passed on for storage on disk or transmission over a network, although sometimes it may be possible to perform the compression using software running on the computer's central processor.

The alternative option is to digitize and compress the video signal using circuitry inside a camera. The purely digital signal (now better described as a data stream) is then passed to the computer via a high speed interface. The *IEEE 1394* interface, often called by its more colourful code name *FireWire*, is commonly used for this purpose, although low quality cameras, typically used for video conferencing, may use a USB connection.

⇨ The expression 'digital video' has acquired several different meanings. It is sometimes used in a general sense to refer to the storage and manipulation of video data in a digital form, but it is also used more specifically to refer to the technology just described, which performs digitization in camera. An emerging consumer-level technology for digital video in this second sense is based on the *DV* standard, where DV stands for... digital video, as does (or did originally) the DV in DVD. We will usually use the term in its general sense; context will indicate any occasions on which it is used in some more restricted sense.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 322
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Digitization in the camera has one great advantage over digitization in the computer. When an analogue signal is transmitted over a cable, even over a short distance, it will inevitably be corrupted, if only to a small extent, by noise. Noise will also creep in when analogue data is stored on magnetic tape. *Composite* video signals, the type normally used in domestic equipment, are also subject to distortion caused by interference between the colour and brightness information, especially when stored on VHS tape. We have all become used to noise and distortion in broadcast television and VHS video, and don't usually consider it objectionable, but these phenomena can reduce the effectiveness of compression. They typically produce small random fluctuations in colour, so that, for example, areas that should be composed of a single colour, and would compress well, will lose their coherence and compress less well. Some video compression techniques are based on the similarity between adjacent frames. Again, the random nature of any signal degradation will undermine this similarity, making compression less effective, and the resulting picture less satisfactory. If digitization is carried out inside the camera, only digital signals, which, as we pointed out in Chapter 2, are resistant to corruption by noise and interference, are transmitted down cables and stored on tape. Compression is applied to a clean signal at its source and so it will be more effective.

The disadvantage of digitization performed in the camera is that the user has no control over it. The data stream produced by a digital video camera must conform to an appropriate standard, such as the DV standard for domestic and semi-professional camcorders, which stipulate the data rate for the data stream, and thus the amount of compression to be applied. Analogue video capture boards and their associated software normally offer the user control over the compression parameters, allowing a trade-off to be made between picture quality and data rate (and hence file size).

Video capture boards can usually not only perform digitization and compression, but also their inverse, decompression and digital to analogue conversion. Devices which compress and decompress signals are known as compressor/decompressors, invariably contracted to *codecs*. Using a hardware codec, it is possible to capture video signals, store them on a computer, and then play them back at full motion to an external video monitor (i.e. a TV set) attached to the video card's output. Although playback to the computer screen is not impossible, most hardware codecs cannot provide

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 323
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

full screen full motion video to the monitor. More importantly for multimedia producers, we cannot know whether our audience will have any hardware codec available, or if so, which one. So, we need a *software codec*: a program that performs the same function as the dedicated hardware codecs we have described, in order to ensure that the audience will be able to play back the video on a computer's ordinary monitor. Generally, software is slower than dedicated hardware, so playing full-screen video at broadcast frame rates using a software codec is only possible on the fastest of desktop computers, and we have to sacrifice many aspects of quality in video that is intended for playback in this way. Generally, compression algorithms that are suitable for software codecs are different from those suitable for hardware implementation, so we will often have to recompress captured video material — resulting in a type of generational loss of quality — in order to prepare it for incorporation in a multimedia production.

Video Standards

Analogue Broadcast Standards

There are three sets of standards in use for analogue broadcast colour television. The oldest of these is *NTSC*, named after the (US) National Television Systems Committee, which designed it. It is used in North America, Japan, Taiwan and parts of the Caribbean and of South America. In most of Western Europe a standard known as *PAL*, which stands for Phase Alternating Line, referring to the way the signal is encoded, is used, but in France *SECAM* (*Séquential Couleur avec Mémoire*, a similar reference to the signal encoding) is preferred. PAL is also used in Australia and New Zealand and in China, while SECAM is used extensively in the former Soviet Union and in Eastern Europe. The standards adopted in Africa and Asia for the most part follow the pattern of European colonial history. The situation in South America is somewhat confused, with NTSC and local variations of PAL being used in different countries there.

The NTSC, PAL and SECAM standards are concerned with technical details of the way colour television pictures are encoded as broadcast signals, but their names are used loosely to refer to other

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 324
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

characteristics associated with them, in particular the frame rate and the number of lines in each frame. To appreciate what these figures refer to, it is necessary to understand how television pictures are displayed.

Like computer monitors, television sets work on a raster scanning principle. Conceptually, the screen is divided into horizontal lines, like the lines of text on a page. In a CRT (cathode ray tube) set, three electron beams, one for each additive primary colour, are emitted and deflected by a magnetic field so that they sweep across the screen, tracing one line, then moving down to trace the next, and so on. Their intensity is modified according to the incoming signal so that the phosphor dots emit an appropriate amount of light when electrons hit them. The picture you see is thus built up from top to bottom as a sequence of horizontal lines. (You can see the lines if you look closely at a large TV screen.) Once again, persistence of vision comes into play, to make this series of lines appear as a single picture.

As we remarked above, the screen must be refreshed about 40 times a second if flickering is to be avoided. Transmitting an entire picture that many times a second requires an amount of bandwidth that was considered impractical at the time the standards were being developed. Instead, each frame is divided into two *fields*, one consisting of the odd-numbered lines of each frame, the other of the even lines. These are transmitted one after the other, so that each frame (still picture) is built up by *interlacing* the fields. The fields are variously known as odd and even, upper and lower, and field 1 and field 2. Originally, the rate at which fields were transmitted was chosen to match the local AC line frequency, so in Western Europe, a field rate of 50 per second, and hence a frame rate of 25 per second, is used for PAL; in North America a field rate of 60 per second was used for black and white transmission, but when a colour signal was added for NTSC, it was found to cause interference with the sound, so the field rate was multiplied by a factor of 1000/1001, giving 59.94 fields per second. Although the NTSC frame rate is often quoted as 30 frames per second, it is actually 29.97.

When video is played back on a computer monitor, it is not generally interlaced. Instead, the lines of each frame are written to a frame buffer from top to bottom, in the obvious way. This is known as *progressive scanning*. Since the whole screen is refreshed from the frame buffer at a high rate, flickering does not occur, and in fact much lower frame rates can be used than those necessary for

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 325
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

broadcast. Fields may be combined into frames when an analogue video signal is digitized, or they may be stored separately and only combined when the material is played back. Since fields are actually separated in time, combining them in this way can cause undesirable effects. If an object is moving rapidly, it may change position between the two fields. Figure 10.2 shows an extreme example of a flash of light on the surface of water. When the fields are combined into a single frame for progressive display, there are discontinuities, since adjacent lines come from different fields. Even where the movement is less rapid, the same effect will occur in a less extreme form: the edges of moving objects will have a comb-like appearance where they are displaced between fields, as shown in Figure 10.3. To prevent this effect, it may be necessary to 'de-interlace' by averaging the two fields when constructing a single frame; this, however, is a relatively poor compromise. Another option is to discard half the fields — say all the even fields — and to interpolate the missing information for the fields that remain, to give full frames. This also gives predictably poor results. On the other hand, the fact that there are 60 or 50 fields per second in analogue video can sometimes usefully be exploited by converting each field into a single frame, thus giving two seconds of slower but fluid video for each second captured. If the resulting video is either played at full speed (30 or 25 fps), or at a small size, the loss of image quality resulting from the interpolation necessary to create frames from fields will scarcely be noticeable.

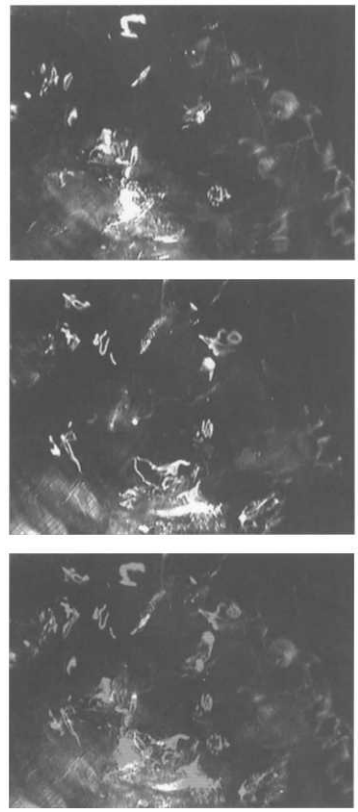


Figure 10.2
Fields and interlaced frame with very rapid change

Each broadcast standard defines a pattern of signals to indicate the start of each line, and a way of encoding the picture information itself within the line. In addition to the lines we can see on the picture, some extra lines are transmitted in each frame, containing synchronization and other information. An NTSC frame contains 525 lines, of which 480 are picture; PAL and SECAM use 625 lines, of which 576 are picture. It is common to quote the number of lines and the field rate together to characterize a particular scanning standard; what we usually call NTSC, for example, is 525/59.94.

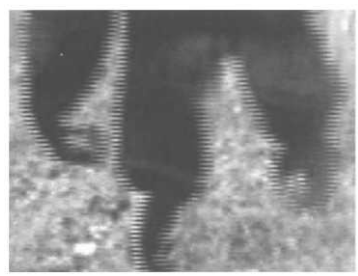


Figure 10.3
Interlacing artefacts

⇨ It is possible that you might need to digitize material that was originally made on film and has been transferred to video tape. This would be the case if you were making a multimedia film guide, for example. Most film footage is projected at 24 frames per second so there is a mismatch with all the video standards. In order to fit 24 film frames into (nearly) 30 NTSC video frames, a stratagem known as *3-2 pulldown* is employed. The first film frame is recorded for the first three video fields, the second for two, the third for three

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 326
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

again, and so on. If you are starting with material in this form, it is best to remove the 3-2 pulldown after it has been digitized (a straightforward operation with professional video editing software) and revert to the original frame rate of 24 per second. Using PAL, films are simply shown slightly too fast, so it is sufficient to adjust the frame rate.[1]

1
'Made for TV' movies are usually shot at the prevailing TV frame rate.

In production and post-production studios, analogue video is represented in *component* form: three separate signals, representing luminance and two colour difference values, are used. As we explained on page 172, the components are said to represent picture information in *YUV colour*. For transmission, the three components are combined into a single *composite* signal, with Y, U and V being transmitted on a single carrier.[2] Each colour difference component (U and V) is allocated half as much bandwidth as the luminance (Y), a form of analogue data compression, which is justified by the empirical observation that human eyes are less sensitive to variations in colour than to variations in brightness. As we remarked above, there is inevitably some interference between the components, especially when a composite signal is stored on tape, although for capture direct from a good camera a composite signal will be perfectly adequate for anything other than broadcast quality productions. A compromise between component video and composite video separates the luminance from the two colour difference (which remain combined into one *chrominance* signal). This representation is employed in high-level consumer and semi-professional video equipment that uses the S-VHS and Hi-8 tape formats; the term *S-video* is used to describe this form of signal. Despite being a compromise solution, the significantly better quality of the video tapes available for recording this signal means that S-video is very widely used for sourcing video footage for low-cost multimedia production; it is also frequently used when 'amateur' footage is to be shown on television, as in public access programming.

2
Actually Y′, U and V.

In practical terms, if you are capturing analogue video to use in a multimedia production, it is necessary to ensure that your capture board is compatible with the signals produced by your camera or VTR. Composite and S-video inputs and outputs are found on most capture cards, but only more expensive professional equipment can handle component signals.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 327
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Digital Video Standards

The standards situation for digital video is no less complex than that for analogue video. This is inevitable, because of the need for backward compatibility with existing equipment — the use of a digital data stream instead of an analogue signal is orthogonal to scanning formats and field rates, so digital video formats must be capable of representing both 625/50 and 525/59.94. The emerging HDTV standards should also be accommodated. Some attempt has been made to unify the two current formats, but unfortunately, different digital standards for consumer use and for professional use and transmission have been adopted.

Like any analogue data, video must be sampled to be converted into a digital form. A standard officially entitled *Rec. ITU-R BT.601*, but more often referred to as *CCIR 601*[3] defines sampling of digital video. Since a video frame is two-dimensional, it must be sampled in both directions. The scan lines provide an obvious vertical arrangement; CCIR 601 defines a horizontal sampling picture format consisting of 720 luminance samples and two sets of 360 colour difference samples per line, irrespective of the scanning standard. Thus, ignoring chrominance and interlacing for a moment, an NTSC frame sampled according to CCIR 601 will consist of 720×480 pixels, while a PAL frame will consist of 720×576 pixels.

3 CCIR was the old name of the organization now known as ITU-R.

Observant readers will find this perplexing, in view of our earlier statement that the sizes of PAL and NTSC frames are 768×576 and 640×480 pixels, respectively, so it is necessary to clarify the situation. PAL and NTSC are analogue standards. Frames are divided vertically into lines, but each line is generated by a continuous signal, it is not really broken into pixels in the way that a digital image is. The value for the number of pixels in a line is produced by taking the number of image lines and multiplying it by the *aspect ratio* (the ratio of width to height) of the frame. This aspect ratio is 4:3 in both PAL and NTSC systems, which gives the sizes originally quoted. Video capture cards which digitize analogue signals typically produce frames in the form of bitmaps with these dimensions. The assumption underlying the calculation is that pixels are square. By relaxing this assumption, CCIR 601 is able to specify a sampling rate that is identical for both systems, and has desirable properties such as providing the same number

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 328
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

of horizontal samples in every line. CCIR 601 pixels, then, are not square: for 625 line systems, they are slightly wider than high, for 525 line systems they are slightly higher than wide. Equipment displaying video that has been sampled according to CCIR 601 must be set up to use pixels of the appropriate shape.

⇨ We have simplified somewhat. There are actually 858 (NTSC) or 864 (PAL) samples per line, but only some of these (the 'active samples') constitute part of the picture. While digital video editing software generally works with 720 pixels per line, one expert on digital video has made the following remark (with reference to 525/59.94 scanning):

> "[...]the debate over the true number of active samples is the trigger for many hair-pulling cat-fights at TV engineering seminars and conventions, so it is healthier to say that the number lies somewhere between 704 and 720. Likewise, only 480 lines out of the 525 lines contain active picture information. Again, the actual number is somewhere between 480 and 496."[4]

4 Chad Fogg, *The MPEG2 FAQ.*

Like component analogue video, video sampled according to CCIR 601 consists of a luminance component and two colour difference components. The colour space is technically $Y'C_BC_R$ (see page 172) — you can consider the three components to be luminance Y, and the differences B−Y and R−Y. We described earlier how analogue video signals are compressed by allocating half as much bandwidth to each colour difference signal as to the luminance. In a similar way, and with the same justification, digital video is compressed by taking only half as many samples for each of the colour difference values as for luminance, a process known as *chrominance sub-sampling.* The arrangement of samples used in CCIR 601 is called *4:2:2* sampling; it is illustrated in Figure 10.4. In each line there are twice as many Y samples as there are samples of each of B−Y and R−Y. Since, for those pixels whose colour is sampled, all three values are sampled at the same point, the samples are said to be *co-sited.* The resulting data rate for CCIR 601 video is, using eight bits for each component, 166 Mbits (just over 20 Mbytes) per second.

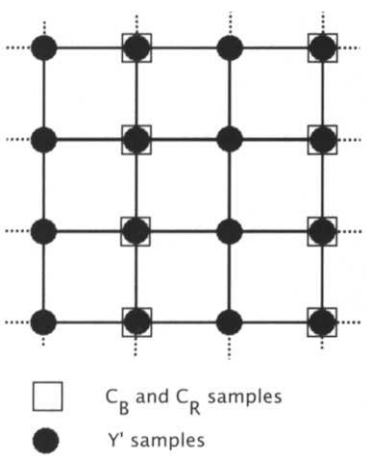


Figure 10.4
4:2:2 sampling

Why 4:2:2 and not 2:1:1? Because other sampling arrangements are possible. In particular, some standards for digital video employ either 4:1:1 sampling, where only every fourth pixel on each line is sampled for colour, or 4:2:0, where the colour values are sub-sampled by a factor of 2 in both the horizontal and vertical

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 329
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

directions[5] — a somewhat more complex process than it might at first appear, because of interlacing. (The notation 4:2:0 is inconsistent; it certainly does not mean that only one of the colour difference values is sampled.)

Sampling produces a digital representation of a video signal. This must be compressed and then formed into a data stream for transmission. Further standards are needed to specify the compression algorithm and the format of the data stream. Two separate standards are emerging, both based on $Y'C_BC_R$ components, scanned according to CCIR 601, but with further chrominance sub-sampling. Digital video equipment intended for consumer and semi-professional use (such as corporate training video production) and for news-gathering is based on the *DV* standard; studio equipment, digital broadcast TV and DVD are based on *MPEG-2*. Neither of these is a single standard, although the situation is not entirely chaotic. The main variations on DV — DVCAM and DVPRO — concern tape formats, and use the same compression algorithm and data stream as DV. There are, however, a high quality DVPRO, and a professional Digital-S format, which use 4:2:2 sampling, unlike DV, which uses 4:1:1. MPEG-2 was designed as a family of standards, organized into different *profiles* and *levels*. Each profile defines a subset of features of the MPEG-2 data stream; as we will see in the next section, MPEG-2 allows some scope for different ways of encoding the digital video data. Each level defines certain parameters, notably the maximum frame size and data rate, and chrominance sub-sampling. Each profile may be implemented at one or more of the levels, although not every combination of level and profile is defined. The most common combination is *Main Profile at Main Level (MP@ML)*, which uses CCIR 601 scanning with 4:2:0 chrominance sub-sampling; it supports a data rate of 15 Megabits per second and allows for the most elaborate representation of compressed data provided by MPEG-2. MP@ML is the format used for digital television broadcasts and for DVD video.

We should emphasize that the standards outlined in the preceding paragraph concern the data streams produced by equipment such as digital video cameras and VTRs. Once the data has entered a computer a whole new range of possibilities is opened up. Likewise, video data that has been digitized by a capture board from an analogue signal is unlikely to conform to either the DV or MPEG-2 standards.

5 In 4:2:0 sub-sampling, the chroma and luminance samples are not co-sited. The chrominance samples are positioned halfway between lines, i.e. they are averaged between two lines.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 330
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Digital video formats are intimately bound up with compression, so before we can consider how video may be represented and manipulated inside a computer we must look at compression techniques that are applied to it.

Introduction to Video Compression

We have seen that video data is usually compressed when it is digitized. The form of compression applied by digital video cameras and video capture boards is usually optimized for playback through the same devices. When we want to use a video sequence as part of a multimedia production, we cannot rely on our users having any particular video hardware, so the only safe assumption is that the video is going to be played back using their computer's processor. Additionally, we have to assume that the video data must be delivered from a hard disk, at best, or from a CD-ROM or DVD, or even over a network connection. We will usually prepare video material using a software codec, to apply a form of compression that is more suited to the capabilities of the hardware that is likely to be used by consumers. Typically, therefore, our video data will need to be compressed twice: first during capture and then again when it is prepared for distribution. To appreciate the implications of this, we need to understand the general features of the compression algorithms employed at both of these stages.

All video compression algorithms operate on digitized video consisting of a sequence of bit-mapped images. There are two ways in which this sequence can be compressed: each individual image can be compressed in isolation, using the techniques introduced in Chapter 5, or sub-sequences of frames can be compressed by only storing the differences between them. These two techniques are usually called *spatial compression* and *temporal compression*, respectively, although the more accurate terms *intra-frame* and *inter-frame* compression are also used, especially in the context of MPEG. Naturally, spatial and temporal compression can be used together.

Since spatial compression is really just image compression applied to a sequence of images, it makes sense to distinguish between lossless and lossy methods — the distinction, and the trade-offs implied, are the same as they are for still images. Generally,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 331
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

lossless methods do not produce sufficiently high compression ratios to reduce video data to manageable proportions, except on synthetically generated material (such as we will consider in Chapter 11). However, although compressing and recompressing video usually leads to a deterioration in image quality, and should be avoided if possible, recompression is often unavoidable, since the compressors used for capture are not suitable for delivery for multimedia. Furthermore, for post-production work, such as the creation of special effects, or even fairly basic corrections to the footage, it is usually necessary to decompress the video so that changes can be made to the individual pixels of each frame. For this reason, it is wise, if you have sufficient disk space, to work with uncompressed video during the post-production phase. That is, once the footage has been captured and selected, decompress it and use uncompressed data while you edit and apply effects, only recompressing the finished product for delivery.

⇨ There is considerable irony in the fact that one of the most loudly voiced advantages of digital video (and audio) is that it suffers no 'generational loss', unlike analogue video, which inevitably degrades every time you copy a tape, as you must whenever you perform any editing, and when you prepare material for distribution. Digital data can be copied as many times as you like, without any loss of quality — provided all you are doing is making an exact copy. As soon as you start decompressing and recompressing between copies, something very much like generational loss, only uglier, happens after all. The irony is compounded by the fact that considerable effort is being expended on devising methods to prevent people making exact copies, for fear of widespread piracy of digital video.

The principle underlying temporal compression algorithms is simple to grasp. Certain frames in a sequence are designated as *key frames*. Often, key frames are specified to occur at regular intervals — every sixth frame, for example — which can be chosen when the compressor is invoked. These key frames are either left uncompressed, or more likely, only spatially compressed. Each of the frames between the key frames is replaced by a *difference frame*, which records only the differences between the frame originally in that position and either the most recent key frame or the preceding frame, depending on the sophistication of the decompressor. For many sequences, the differences will only affect a small part of the frame. Consider, for example, a typical 'talking head' shot, of a person's head and upper body against a background, such as you might see on a news report. Most of the time, only

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 332
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

parts of the speaker's face will be moving; the background, and perhaps the speaker's upper body, will remain static, so the pixels corresponding to these elements will keep the same values from frame to frame. Therefore, each difference frame will have much less information in it than a complete frame. This information can be stored in much less space than is required for the complete frame.

You will notice that we have described these compression techniques in terms of frames. This is because we are normally going to be concerned with video intended for progressively scanned playback on a computer. However, the techniques described can be equally well applied to fields of interlaced video; while this is somewhat more complex, it is conceptually no different.

Compression and decompression of a piece of video need not take the same time. If they do, the codec is said to be *symmetrical*, otherwise it is *asymmetrical*. In theory, this asymmetry could be in either direction, but generally it is taken to mean that compression takes longer — sometimes much longer — than decompression. This is acceptable, except during capture, but since playback must take place at a reasonably fast frame rate, codecs which take much longer to decompress video than to compress it are essentially useless.

⇨ The quality of lossily compressed video can only be judged subjectively — we can measure how much information has been discarded, but not whether it matters to the viewer. Consequently, any terms used to describe video quality are vague at best. One of the most commonly employed terms is 'broadcast quality', which does not mean the quality you receive on a TV set, it means good enough *to be* broadcast or used in post-production — a much higher quality, usually only achieved in a professional studio, using high end equipment. 'Near broadcast quality', a favourite term of marketing people, means 'not broadcast quality'. 'VHS quality' is a vague term, as VHS tape is available in a very wide range of grades, from the cheapest commonly used in domestic VCRs to tapes described as 'professional' or 'broadcast'. Generally, though, it is reasonable to take 'VHS quality' to mean 'just about acceptable'.

Motion JPEG and DV

At present, the most popular approach to compressing analogue video during capture is to apply JPEG compression to each frame,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 333
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

with no temporal compression. JPEG compression is applied to the three components of a colour image separately, and works the same way irrespective of the colour space used to store image data. Analogue video data is usually stored using $Y'C_BC_R$ colour, with chrominance sub-sampling, as we have seen; JPEG compression can be applied directly to this data, taking advantage of the compression already achieved by this sub-sampling.

The technique of compressing video sequences by applying JPEG compression to each frame is referred to as *motion JPEG* or *MJPEG* compression[6], although you should be aware that, whereas JPEG is a standard, MJPEG is only a loosely defined way of referring to this type of video compression. Special-purpose hardware is usually employed, allowing the complex computations required by the JPEG algorithm to be carried out fast enough to keep up with the incoming video signal. Individual hardware manufacturers have implemented motion JPEG independently, and in the absence of any standard, implementations differ slightly, especially in the way that the compressed data is stored. Recently, a consortium of digital video companies has agreed a standard format known as *MJPEG-A*, which is supported by QuickTime (see below), ensuring that compressed data can be exchanged between systems using different hardware.

6 Not to be confused with MPEG.

Like still-image JPEG, motion JPEG codecs allow the user to specify a quality setting, trading off high compression against image quality. Instead of specifying the quality directly, video codecs often allow the specification of a maximum data rate, from which a suitable quality setting is deduced. This is appropriate, for example, if you wish your video to be played back from a DVD, where the maximum data rate is known in advance.

Data rates of around 3 Mbytes per second, corresponding to a compression ratio of around 7:1 are commonly achieved by low- to mid-range capture cards. More expensive cards offer higher data rates (lower compression ratios) and therefore better quality.[7] However, for multimedia use, it is invariably necessary to apply a high degree of compression to the video data for delivery, so although the principle of starting with the highest possible quality material is a sound one, it may not be worthwhile working with the best capture boards if the final product is destined for CD-ROM or the Internet.

7 This is perhaps counter-intuitive: the more you pay for compression, the less of it you get.

Provided the device on which the compressed video data is stored is capable of delivering it at the rate required by the card, MJPEG

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 334
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

video can be played back at full frame size at standard frame rates. Such 'full-frame, full-motion' playback cannot presently be achieved without the aid of special-purpose hardware.

Although in multimedia it is most common to use MJPEG hardware to compress video as it is captured, sometimes we will want to go in the opposite direction, and use software to perform motion JPEG compression so that a video sequence that is already in digital form can be played back through a hardware codec. MJPEG compression is also sometimes used as an archiving format for video, since it provides higher quality than the codecs that are needed to compress material for delivery. However, since JPEG compression is lossy, so is MJPEG compression. This means that whenever video is compressed using MJPEG, some information, and hence picture quality, is immediately lost. Additionally, any subsequent decompression and recompression will exacerbate this loss of quality. It is therefore preferable, if at all possible, to archive uncompressed video, and only to use MJPEG where it is necessary — at the point of capture.

DV equipment uses a compression algorithm similar to MJPEG, though it is not identical. DV is chrominance sub-sampled to 4:1:1. Like JPEG, DV compression is based on the Discrete Cosine Transform (see Chapter 5); in addition it performs some temporal compression, but only between the two fields of each frame. It uses motion compensation, which will be described under MPEG shortly, to help achieve this. Because of the demands of DV-based VTRs, DV must maintain a constant data rate of 25 Mbits (3.25 Mbytes) per second, a compression ratio of 6.6:1 relative to CCIR 601, or, to give a better idea of how much compression is really being achieved by the algorithm, 5:1 relative to the 4:1:1 sampled data stream. The quality is varied dynamically to maintain this rate; this means that frames with relatively little motion will be spatially compressed less than, and hence will be of a higher quality than, those with more motion, because in the latter case less temporal compression will be possible. DV data should be of better quality than MJPEG compressed analogue video that has been compressed by a capture board to a similar compression ratio, because of the absence of noise, which, as we stated earlier, interferes with the effectiveness of compression.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 335
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Software Codecs for Multimedia

Among the many software codecs that have been developed, four are presently considered especially suitable for compressing video destined for delivery on CD-ROM or (with a little patience) over the Internet. The most elaborate of these, MPEG-1, will be described in the next section. The other three are called *Cinepak*, *Intel Indeo*, and *Sorenson*.

All three are based on a technique known as *vector quantization*, which works in the following manner: each frame is divided up into small rectangular blocks of pixels, the 'vectors' in vector quantization. The codec uses a collection of constant vectors, known as a *code book*; the code book vectors represent typical patterns that might occur in an image, such as flat areas of colour, sharp or soft edges, or different textures. Quantization is the process of allocating each vector in the image to the vector in the code book which approximates it most closely. (It is a generalization of scalar quantization, familiar from our description of digitization, when a value from some continuous range is approximated by one of a number of fixed levels.) Vector quantization provides compression, because each vector in the image can be replaced by an index into the code book. The image can be reconstructed from these indices and the code book by a simple process of putting together copies of the code book vectors corresponding to the stored indices. Decompression is thus very efficient, and can be carried out without special-purpose hardware. Compression, on the other hand, is a computationally intensive process, so these codecs are highly asymmetrical: compressing a frame using Cinepak, for example, typically takes more than 150 times as long as decompressing it.

Cinepak, Intel Indeo and Sorenson all augment their vector quantization compression with temporal compression using key frames and difference frames. The first two use a straightforward differencing scheme, while Sorenson uses a more sophisticated scheme, including motion compensation, which is closer to that used in MPEG compression.

Even though the decompression process is relatively efficient, full-motion full-screen playback cannot be achieved on mid-range processors with these codecs. Therefore, they are usually applied to small frames to be played back at reduced frame rates.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 336
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Respectable (VHS quality) images can be achieved at quarter-frame size (320×240 pixels) and 12 frames per second — certainly not good enough for the discerning viewer, but acceptable for many purposes. The Sorenson codec (the most efficient of the three) can compress video with these parameters so that it has a data rate of only 50 kilobytes per second, which is well within the capabilities of a 'multimedia PC' and can even be delivered by a 1× speed CD-ROM drive. (It should be borne in mind, though, that this data rate is still well in excess of that of an ISDN line, let alone a modem, so there is no chance of using these codecs to achieve real-time playback over a dial-up Internet connection. They could be used in this way over a 10 base T ethernet, though.)

Of the three, Cinepak has been established the longest. It remains popular, because of the high compression ratios it can achieve, and the efficiency with which it plays back compressed material, which makes it especially suitable for use on older computers. Intel's Indeo is similar in its general characteristics, but is less asymmetrical, being roughly 30% faster than Cinepak at compressing. It was originally intended for hardware implementation, but is now available in an efficient software version. It is generally held that Cinepak is superior for material with a lot of motion, but Indeo is better for more static material, for which it does a better job of preserving colours accurately. Sorenson is the most recently produced of these three codecs, and is widely hyped as being superior to the other two, both in quality and in the compression ratios it can achieve. It is, however, only available with QuickTime 3.0 or higher.[8]

8
The version of the Sorenson compressor distributed free with QuickTime is a basic one that lacks some of the features that make the full version attractive. The full version is relatively expensive.

MPEG Video

Although all of the codecs described so far are widely available, none of them is anything more than a *de facto* standard. Several international standards for compressed digital video are grouped together under the name *MPEG*, which stands for Motion Picture Experts Group — the resemblance to JPEG is not coincidental.

Earlier in this chapter, we briefly described the MPEG-2 format for broadcast digital video. For video to be incorporated in multimedia, the earlier MPEG-1 format is more relevant, being designed for lower data rates and progressive rendering. An emerging MPEG-4 standard provides support for integrated multimedia, including

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 337
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

video, at data rates ranging from as low as 5 kbits per second to around 4 Mbytes per second. It is to be expected that this ambitious standard will have a growing importance for distributed multimedia, but at the time of writing, it is not published and no implementations based on it exist, so we will concentrate on MPEG-1 — in this chapter, its video aspects only.

The MPEG-1 standard[9] doesn't actually define a compression algorithm: it defines a data stream syntax and a decompressor, allowing manufacturers to develop different compressors, thereby leaving scope for 'competitive advantage in the marketplace'. In practice, the compressor is fairly thoroughly defined implicitly, so we can describe MPEG-1 compression, which combines temporal compression based on motion compensation with spatial compression based, like JPEG, on quantization and coding of frequency coefficients produced by a discrete cosine transformation of the data.

A naïve approach to temporal compression consists of subtracting the value of each pixel in a frame from the corresponding pixel in the previous frame, producing a difference frame. In areas of the picture where there is no change between frames, the result of this subtraction will be zero. If change is localized, difference frames will contain large numbers of zero pixels, and so they will compress well — much better than a key frame. Often, though, we may be able to do better, because pictures are composed of objects that move as a whole: a person might walk along a street, a football might be kicked, or the camera might pan across a landscape with trees. Figure 10.5 shows a simple example, consisting of an object — actually the lower body and legs of a gibbon, swinging from a rope — moving through two consecutive frames of a movie. As the dark shape of the gibbon moves across the screen, it obscures different parts of the vegetation behind it. Figure 10.6 shows the area within which pixels may have changed. If we could somehow identify the coherent area corresponding to the animal, we would only need to record its movement together with the changed pixels in the area shown in Figure 10.7. *Motion compensation* is an attempt to do this.

MPEG compressors do not attempt to identify objects in a scene. Instead, they divide each frame into blocks of 16×16 pixels known as *macroblocks* (to distinguish them from the smaller blocks used in the DCT phase of compression), and attempt to predict the whereabouts of the corresponding macroblock in the next frame. No high-powered artificial intelligence is used in this prediction: all possible displacements within a limited range are tried, and the

9 ISO/IEC 11172: 'Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbit/s'



Figure 10.5
An object moving between frames



Figure 10.6
Area of potential change

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 338
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

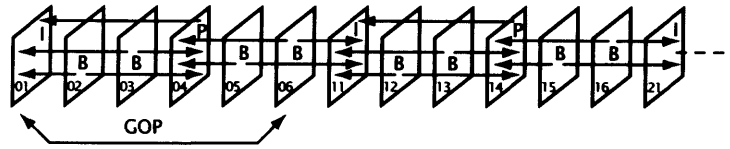


Figure 10.9
An MPEG sequence in display order

best match is chosen.[10] The difference frame is then constructed by subtracting each macroblock from its predicted counterpart, which should result in fewer non-zero pixels, and a smaller difference frame after spatial compression. The price to be paid is that, in addition to the difference frame, we now have to keep the motion vectors describing the predicted displacement of macroblocks between frames.

Temporal compression has to start somewhere. MPEG key frames are called *I-pictures*, where I stands for *intra*. These frames are purely spatially (intra-frame) compressed. Difference frames that use previous frames are called *P-pictures*, or 'predictive pictures'. P-pictures can be based on an earlier I-picture or P-picture. MPEG goes further and allows for frames that are predicted from later frames; these are *B-pictures*. Figure 10.8 shows why backward prediction can be useful. In this frame, which precedes those shown in Figure 10.5, the tree trunk on the right of the picture is hidden. If we wished to use this frame as an I-picture, the following P-picture would have to record all the pixels of the tree trunk. However, the trunk is present in the following frame (the lower one in Figure 10.5), so if the middle frame is predicted from the one before itself and the one after, further compression will be achieved. B-pictures can use motion compensation from the next I- or P-pictures, or both, hence their full name 'bi-directionally predictive' pictures.

A video clip can be encoded in compressed form as a sequence of I-, P- and B-pictures. It is not a requirement that this sequence be regular, but encoders typically use a repeating sequence, known as a *Group of Pictures* or *GOP*, which always begins with an I-picture. Figure 10.9 shows a typical example.[11] The GOP sequence is IBBPBB; the picture shows two such groups: frames 01 to 06 and 11 to 16. The arrows indicate the forward and bi-directional prediction. For example, the P-picture 04 depends on the I-picture 01 at the start of its GOP; the B-pictures 05 and 06 depend on the preceding P-picture 04 and the following I-picture 11. All three types of picture are compressed using the MPEG version of JPEG

10
Extensive searching is not a good idea, since it implies the need for larger displacement vectors.



Figure 10.7
Area of potential change, allowing for motion of the object



Figure 10.8
I-picture with a hidden object

11
Read it from left to right, like a time line.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 339
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

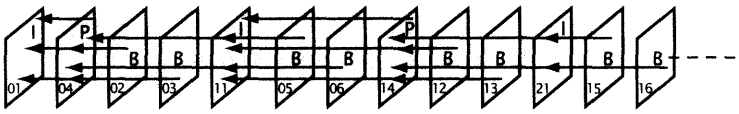


Figure 10.10
An MPEG sequence in bitstream order

compression. Published measurements[12] indicate that, typically, P-pictures compress three times as much as I-pictures, and B-pictures one and a half times as much as P-pictures. However, reconstructing B-pictures is more complex than reconstructing the other types, so there is a trade-off to be made between compression and computational complexity when choosing the pattern of a GOP. An additional factor is that random access to frames corresponding to B- and P-pictures is difficult, so it is customary to include I-pictures sufficiently often to allow random access to several frames each second. Popular GOP patterns include IBBPBBPBB and IBBPBBPBBPBB. However, as we remarked, the MPEG specification does not require the sequence of pictures to form a regular pattern, and sophisticated encoders will adjust the frequency of I-pictures in response to the nature of the video stream being compressed.

12
In [Tud95].

For the decoder, there is an obvious problem with B-pictures: some of the information required to reconstruct the corresponding frame is contained in an I- or P-picture that comes later in the sequence. This problem is solved by reordering the sequence. The sequence of pictures corresponding to the actual order of frames is said to be in 'display order'; it must be rearranged into a suitable 'bitstream order' for transmission. Figure 10.10 shows the bitstream order of the sequence shown in display order in Figure 10.9. All the arrows showing prediction now run from right to left, i.e. every predicted frame comes later in the sequence than the pictures it depends on.[13] You will notice that the first GOP is reordered differently from the second; any subsequent groups will extend the pattern established by the second.

13
For the B-pictures, we have run the arrows to the relevant P- and I-pictures together, with an intermediate arrowhead, in an attempt to keep the diagram less cluttered.

Before any of this compression is done, MPEG video data is chroma sub-sampled to 4:2:0. If, in addition to this, the frame size is restricted to 352×240[14], video at a frame rate of 30 fps can be compressed to a data rate of 1.86 Mbits per second — the data rate specified for compact disc video. This is the typical format for MPEG-1 video, although it can be used with larger frame sizes and other frame rates. MPEG-1 cannot, however, handle interlacing or HDTV formats, hence the need for MPEG-2 for broadcasting and studio work. For multimedia, however, MPEG-1 is an excellent

14
4:2:0 video of this size is said to be in *Source Input Format (SIF)*.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 340
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

form of compression that provides good quality compared to the software codecs described in the previous section.

The preceding description should have made it clear that MPEG compression and decompression are computationally expensive tasks — and there are further complications which we have glossed over. Initially, MPEG video, like MJPEG, could only be played back using dedicated hardware. Indeed, the parameters used for CD video were chosen largely so that MPEG decoders could be accommodated in VLSI chips at the time the standard was drawn up (1993). Advances in processor speed mean that it has since become feasible to play back MPEG at only slightly lower frame rates and sizes using software only — a G3 PowerMac takes 320×240 at 24 fps in its stride, producing smooth VHS quality playback. The associated data rate of 264 kbytes per second can be delivered by a 2× speed CD-ROM drive. File sizes are by no means small, however. A 650 Mbyte CD-ROM will only hold just over 40 minutes of video at that rate; an 8.75 Gbyte DVD has room for over nine hours. (You would only use MPEG-1 on DVD if your video was part of a multimedia production, though. DVDs employ MPEG-2 when they are Digital *Video* Disks.)

QuickTime

As preceding sections have shown, there is a plethora of digital video compression schemes and data formats in existence: DV, MPEG-1 and MPEG-2, several varieties of MJPEG, various software compressors, including the ones described earlier, proprietary schemes devised by particular manufacturers of digital video workstations, not forgetting uncompressed video. Each of these schemes requires the information to be encoded in a different way; within each encoding there is scope for defining incompatible file formats — even some standards, such as the MPEGs, only define a data stream, not a file format. Because of the diversity of requirements, there is little hope of being able to design a universal video file format, and the usual political considerations make it unlikely that all the parties involved would agree to it, even if one could be devised. A more profitable approach to standardization is to base it on an architectural framework, defined at a sufficiently abstract level to accommodate a multiplicity of concrete video representations. Several such approaches have been proposed, but *QuickTime* has established itself as a *de facto* standard.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 341
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

QuickTime was introduced by Apple in 1991, and reached a mature state with the release of version 3.0 in mid-1998.[15] The objects that QuickTime manipulates are *movies*. For the present, you can consider a movie to be an abstraction of a video sequence, although, in fact, QuickTime movies can contain other types of media as well. Although we speak of a movie containing data, the movie itself is really just a framework for organizing, accessing and manipulating the data that represent the actual video frames, which may be stored separately from the movie itself.

15 In 1999, QuickTime 4.0 added support for streaming, which will be described at the end of this chapter.

Originally, QuickTime's focus was on the temporal aspects of time-based media, including video. Every movie has a *time base*, which records the rate at which it should be played back and its current position; both are specified relative to a time coordinate system which allows the time base to be synchronized to a clock so that movies can be played back at the correct speed on any system. When a movie is played, if the frames cannot be displayed fast enough to maintain the required frame rate, some are dropped so that the overall duration of the movie will be correct and synchronization (for example with an associated sound track) will be maintained. The time coordinate system also makes it possible to identify specific points in a movie, so that individual frames can be accessed; this in turn facilitates non-linear editing of movies — a topic which will be described in the next section.

The success of QuickTime is largely due to its being a component-based architecture. This means that it is possible to plug components (small programs conforming to certain interface conventions) into the QuickTime structure in order to deal with new formats and provide new implementations of operations. A set of standard components is supplied as part of the distribution. These include a number of *compressor components*, implementing Sorenson, Cinepak, and Intel Indeo codecs, as well as several others developed for specific tasks such as compressing computer-generated animation. *Sequence grabber components* are used to digitize video. They use lower level components to obtain any necessary parameters from the user, and to communicate with digitizing hardware. A standard *movie controller* component is used to provide a user interface for playing movies; it is illustrated in Figure 10.11, where you can see that the control buttons are based on those normally found on domestic VCRs, so that their use is obvious to most people. There are also components called *transcoders*, which translate data between formats that use the same



Figure 10.11
A standard QuickTime movie controller

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 342
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

compression algorithm, without the need for decompression and recompression.

QuickTime does have its own file format, which provides a very flexible way of storing video and other media.[16] To avoid imposing this format on everybody, and to enable application software based on QuickTime to access other types of file, components have been added to make it possible to manipulate files in other formats as if they were native QuickTime. Formats supported in this way include MPEG-1 and DV, OMF (a high-end professional format), and Microsoft's AVI, and its extension OpenDML. As this demonstrates, QuickTime's component architecture makes it easily extensible. Another common example occurs when a new video capture card is developed, with its own features and idiosyncrasies. Provided its manufacturer writes a (relatively simple) *video digitizer component*, any application program based on QuickTime can capture and play back video using the new card, edit that video, and convert it to other formats. Looking at this from the other side, anybody who devises a new video format or codec does not have to implement all the high level editing operations that are necessary to make it useful — QuickTime provides all that, using the interface supplied by the components associated with the type of data in question. This abstraction away from specific formats is passed on to end users of video editing and playback software. For example, when a video editor wishes to save an edited movie, he or she is presented with options for selecting a compression method. If a new codec is added to their system, it will show up as a new alternative within the same interface.

16
The QuickTime file format has also been adopted as the file format for storing MPEG-4 data.

QuickTime 4.0 is available in a fully compatible form on the 32-bit Windows platforms as well as MacOS, and QuickTime is also supported on SGI workstations for professional use. A version of QuickTime for Java has also been developed. As well as the basic functionality implemented in the QuickTime system software, a plug-in for popular Web browsers is distributed with QuickTime, which means that QuickTime files can be used as a format for distributing video over the World Wide Web.

The only other formats that you might encounter or consider for use in multimedia are MPEG and AVI. We have stated that MPEG does not define a file format, but there is a rather obvious way of storing an MPEG data stream in a file: just record the bytes in the order they appear in the stream. Files of this form are becoming quite common.[17] They can be played back by QuickTime.

17
Often they are identified by the extension .mpg.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 343
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

So can AVI files, although they are the native format for Microsoft's Video for Windows, a system with similar objectives to QuickTime. Because of the pervasiveness of Windows operating systems, AVI has been widely used, but Video for Windows is generally held to be technically inferior to QuickTime, and has indeed been abandoned by Microsoft, whose attempts at a successor have been overshadowed by the adoption of QuickTime as a *de facto* cross-platform standard. As we stated earlier, QuickTime 3.0 and later versions can handle AVI files directly.

Digital Video Editing and Post-Production

Shooting and recording video only provides raw material. Creating a finished piece of video — whether it is a feature film or a small clip for a Web site — requires additional work. *Editing* is the process of making a constructed whole from a collection of parts. It comprises the selection, trimming and organization of the raw footage, and where sound is used, its combination with the picture. *Transitions*, such as dissolves, may be applied between shots, but no changes are made to the footage itself. We contrast this with *post-production*, which is concerned with making changes or adding to the material. Many of the changes made at this stage are generalizations of the image manipulation operations we described in Chapter 5: colour and contrast corrections, blurring or sharpening, and so on. Compositing — the combination or overlaying of elements from different shots into one composite sequence — is often carried out during post-production; for example, figures may be inserted into background scenes that were shot separately. Elements may be animated during post-production, and animation may be combined with live action, in the manner that has become characteristic of film special effects.

People have been making films for over a hundred years, and during that time an elaborate set of conventions about how film — and, by extension, video — is edited have developed. For example, action sequences in Hollywood films are typically cut 'on the action'; that is, cuts are made while something is happening, in order to distract the viewer's attention from the cut itself. In contrast, flashback sequences are often introduced by a long dissolve, typically starting from a shot of a person starting to tell a story about earlier events,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 344
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

with the narration continuing over the first shot of the flashback. Here, the transition is used to signify the change of time frame, and is supposed to be evident to the viewer, or this method of signification will not work. Viewers expect film to be edited in accordance with conventions such as these; if it is not, as in certain types of avant garde work, then that fact in itself makes a statement. Thus, in contrast to Hollywood directors, some of the French *nouvelle vague* directors of the 1960s deliberately used 'jump cuts', occurring apparently gratuitously in the middle of a scene, to draw attention to the constructed nature of their films, and cut straight to flashbacks instead of using dissolves, acknowledging that film is only seen in its own time. This kind of editing requires the audience to be more alert — to work harder — if they are to appreciate what the film-maker is doing (or even follow the narrative, in some cases).[18] Early film makers, such as Dziga Vertov and Sergei Eisenstein, experimented with editing as much as with the process of shooting film, and their work is still referred to by students of film today.

18
It is interesting to observe that when Woody Allen, working within the commercial milieu, used similar devices in films such as *Husbands and Wives*, they were dismissed by mainstream critics as being intrusive and distracting.

All film and video is constructed. A common example is the way in which a conversation between two people facing each other is conventionally presented: each person occupies the field of view while they speak; when the speaker changes, a cut occurs to a 'reverse shot', with the camera apparently moving instantaneously through nearly 180° to face the new speaker — as if the camera were each person in turn, watching the other speak. You have probably seen this so often that you barely think about it, but, in fact, the two actors in the scene were probably not carrying on a conversation in real time while the dialogue was shot; they may not even have been filmed speaking their lines on the same day. The conversation is created in the editing, on the basis of a convention that, if you do stop to think about it, does not resemble the way anybody could see that conversation if it really took place. This is a particularly blatant example of the artifice involved in constructing sequences of moving pictures, but some artifice is always involved — even the picture from a live Webcam is framed in a certain way. While most Hollywood film makers seek to make the construction of a film invisible, and concentrate on telling a story, other directors, such as Jean-Luc Godard, prefer to expose the mechanics of film making, and still others, such as Eric Rohmer, prefer to minimize their reliance on convention, and shoot and edit in a more realistic way, using third person viewpoints and extended shots for dialogue

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 345
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

scenes, with the sound recorded live, for example. But, however a piece of film or video is to be constructed, it will be necessary both to shoot footage and to edit it.

Even if nobody had ever wanted to display video on a computer, incorporate it into a multimedia production, or broadcast it digitally, video would have been digitized, because the advantages of non-linear editing which digitization brings are too compelling to resist. To appreciate this, and to understand the metaphors used by digital editing systems, you have to consider traditional methods of film and video editing.

Film and Video Editing

Editing film is a physical process. To a film editor, a 'cut' is not merely a figurative reference to an edit that produces a discontinuity in the flow of pictures, it is a real severing of the film itself, which divides a strip of film into two clips which may then be spliced together with others to compose a scene. Clips can be trimmed, cut and spliced, arranged and rearranged, for as long as the fabric of the film holds up. The main problem with this basic editing is keeping track of everything — a good deal of labelling and hanging strips of film from hooks is called for.

Although making straight cuts in film is straightforward, creating other types of transition between clips is much less so, and requires special equipment. Suppose, for example, you wanted a scene of someone falling asleep to dissolve into a dream sequence. Such an effect is usually achieved using a device called an *optical printer*, which is a rig that directs the light from a pair of projectors into a camera.[19] Optical filters and masks can be interposed to control the amount of light from each projector reaching the camera. For our dissolve, we would put the end of the falling asleep scene on one projector and the beginning of the dream sequence on the other; initially, filters would be set up so that all the light from the first projector and none from the second reached the camera. Gradually, the filters would be adjusted, so that the two beams were mixed, with the proportion from the second projector increasing as that from the first decreased. The result would be a piece of film with the required dissolve effect; this would then be spliced in place of the corresponding frames of the two original clips.

19
There are different types of optical printer. Some only use a single projector, but expose the film in the camera more than once, to achieve the same effects as the set-up described here.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 346
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Despite the apparent simplicity of the set-up, exceptionally sophisticated effects can be achieved using such 'opticals', in conjunction with techniques such as matte painting or with models.[20] One drawback is that opticals are usually done by a specialist laboratory, so the film editor cannot actually see what the transition looks like until the film has been developed. This leaves little room for experimentation. It is no coincidence that the straight cut forms the basis of most films' structure.

20
The effects in the originally released version of *Star Wars* were purely optical, not digital.

Traditional video editing, although the same as film editing in principle, is quite different in practice. It is virtually impossible to cut video tape accurately, or splice it together, without destroying it. The only way to rearrange pictures recorded on video tape was to copy them onto a new tape in the desired order. If, for example, we wanted to cut from an outside view of a house to a scene in a room inside it, and had first recorded interior scenes and then added some outside shots to the end of the same tape, we could use two tape machines (call them A and B) to assemble the material in the required order. The output signal from machine A is fed to the input of machine B, so that B records what A plays. The original tape is put on machine A and wound forward to the point at which the desired part of the exterior scene begins, then B is started and records that scene. Recording is stopped while machine A is rewound to the exact beginning of the interior, which is then copied onto the tape in machine B, which now holds the required material in the right order.

A more powerful arrangement is to use three machines, designated A, B and C. If you can arrange that your two scenes are on separate tapes, you load one on to each of A and B, and set the *in point* (first frame) and *out point* (last frame) of each. A device known as an *edit controller* can now start machine A at the in point of the first scene, copying it on to machine C until it reaches its out point, when machine B is started at its in point, adding the second scene to the tape on machine C. One virtue of this arrangement is that the cut can be previewed, by running machines A and B without actually recording on C, so it is possible to experiment with adjusting in and out points until a perfect cut is obtained. The signals can be mixed and modified electronically, so this arrangement also provides a video equivalent of the optical printer, for some standard operations. A rich variety of transitions can be produced this way, and, unlike film transitions, they can be reviewed straight away, and parameters such as the speed of a dissolve can be controlled in

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 347
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

real time. Three machines also make insert editing straightforward. Suppose, for example, that part of a scene has had to be reshot after a first edit of the scene has been done. The original edit can be played off one machine, until the point where the defective footage occurs, at which point the signal is taken from the other machine, and then switched back after the replacement shot has been inserted.

Note that these operations require that the machines be able to start and stop in precisely the right place,[21] which not only requires extremely accurate control mechanisms, but also some means of identifying positions on tapes. *Timecode* is used for this purpose. There are several standards in use, but the only one of any importance is SMPTE timecode. A timecode value consists of four pairs of digits, separated by colons, such as 01:14:35:06, representing hours, minutes, seconds, and frames, so that the complete value identifies a precise frame. A trivially obvious scheme, you may well think — the tricky bit is writing the code onto the tape so that its current frame can be read by a machine. Standards for doing so are in place, and so 'frame-accurate' positioning of tape is possible.

21
It is physically impossible to start a tape transport instantaneously, so some *pre-roll* is always required as the machine comes up to speed.

⇨ Timecode behaves differently depending on the frame rate: for a PAL system, the final component (which identifies the frame number) ranges from 0 to 24, for NTSC it ranges from 0 to 29, but not in the obvious way, because the NTSC frame rate is 29.97. Since there is not an exact number of NTSC frames in a second, SMPTE timecode, which must use exactly 30, drifts with respect to the elapsed time. The expedient adopted to work round this is called *drop frame timecode*, in which frames 00:00 and 00:01 are omitted at the start of every minute except the tenth. (It's a bit like a leap year.) So your count jumps from, say, 00:00:59:29 to 00:01:00:02, but runs smoothly from 00:09:59:29 through 00:10:00:00 to 00:10:00:01. Whether or not it handles drop frame timecode correctly is one measure of how professional a digital video editing program is.

Editing video on two or three machines requires tapes to be copied every time a change is made, and it is inevitable that there will be generational loss as noise is introduced by the copying process. Even the best analogue systems used for commercial video always introduce some loss every time a copy is made. With VHS tape, just two copying operations is usually sufficient to produce serious loss of quality. A second drawback is that the final tape has to

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 348
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

be constructed linearly, from beginning to end, in contrast to film, which can be spliced together in any order. This imposes a less flexible way of working, which inevitably has implications for the nature and quality of video editing.

Digital Video Editing

The unique visual qualities of film might make the process of cutting and splicing worthwhile, but video is such a poor medium that it offers little return for the hardships that must be endured in a three machine edit suite. Digitization has opened the way to a different mode of working, that brings video editing closer in kind to film editing, but without the physical process. An imperfect, but useful, analogy of the difference between linear analogue and non-linear digital video editing is the difference between writing with a typewriter and using a word processor. On a traditional typewriter, words have to be written in their final order, with the potential for corrections limited to what can be achieved with Tipp-Ex, unless an entire sheet is thrown away and retyped — which might upset subsequent pagination. In the latter case, corrections can be made anywhere, text can be composed in any order, without regard to pagination or layout. The ability to randomly access and change data, which is fundamental to the way we build computers, engenders these capabilities in text processing software, and analogous ones in video editing software.

Besides random access, digital video editing's other big advantage is that it is non-destructive. Source clips are never changed. This means that it is possible to cut and recut, potentially forever, as the editor changes his or her mind. Furthermore, in contrast to film, edited digital video can be played back as soon as the hardware on which it is being edited allows. With top-end equipment, this is instantaneously; on desktop machines, there is usually a delay — sometimes a considerable delay — but the delays are measured in minutes and hours, not the days that it may take for film to be processed. Within a few years, at most, even desktop equipment and software will be providing instant playback of edited digital video.

⇨ Despite these advantages, it has been observed by film and video editors who have adopted digital technology that it does not save time. Editors tend to be perfectionists, and the opportunity to experiment with different cuts and transitions can lead to

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 349
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

considerable amounts of time being spent trying to achieve the perfect edit.

Developments in hardware, including DV and FireWire and the increased speed of processors and capacity of hard disks, have led to a broadening of interest in digital video editing. Formerly, only a few professional systems were available, but there is now a range from basic consumer-oriented editors, designed for editing home videos, up to the state of the art editing suites used by film and television studios. Adobe Premiere, which could be described as a mid-range video editing application that incorporates some post-production facilities, is the most widely used video application on desktop platforms, so we will use it as a concrete example,[22] but all video editing programs work in roughly the same way, using the same metaphors derived from traditional film and video editing practices.

22 Note that our descriptions refer to Premiere 5. Earlier releases use a somewhat different interface.

Premiere's interface can be customized to fit different ways of working, but, for most people, editing takes place in three main windows. This is a major difference between video editing and image, graphics, or text editing, where a single document window suffices; the more complex, time-based, nature of video requires that the user be able to view the material in several different ways at once. As we will see in later chapters, other time-based media and multimedia applications have the same requirement for multiple views.

Premiere's three main windows are the project, timeline, and monitor. The project window is used to collect the raw material for a project: video and audio clips, and usually, still images. Video clips may be captured within Premiere from tape or camera, or they may be imported from disk, having been previously captured, possibly with some other application. Within the project window, each clip may be displayed as an icon, with its name, duration, codec, and other information. Related clips can be organized into bins — rather as files are organized in sub-directories — and ordered in whatever way seems perspicuous to the user.

The timeline window provides a visual display of the linear extent of the completed movie, showing the order of its component clips. A simple example is shown in Figure 10.12. Time increases from left to right, and is linearly related to the length of segments of the timeline, so the movie shown here begins with `clip1`, followed by `clip2`, which is roughly twice as long, then the short `clip3`

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 350
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

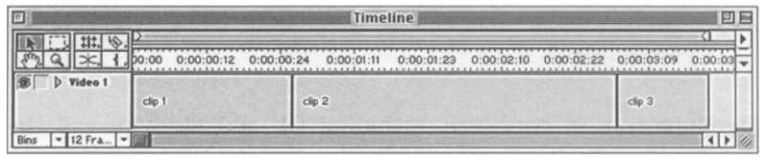


Figure 10.12
A simple composition on the timeline in Premiere

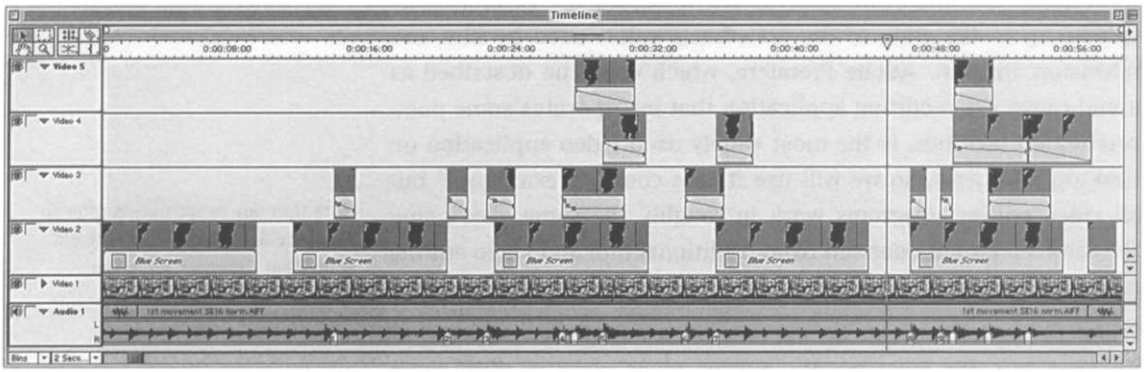


Figure 10.13
A complex Premiere composition

concludes it. In general, timelines may have several video *tracks*, as well as audio tracks. Figure 10.13 shows an example of a complex composition with a sound track. The multiple tracks are used for transitions and overlays, as we will describe shortly.

The third window, the monitor, is where you actually see your picture. (In the other two windows you only see iconic or text representations of clips.) In Premiere 5, the monitor is split into two panes, as shown in Figure 10.14, each with separate playback controls, including jog and shuttle, for stepping through a frame at a time, or scrubbing forwards and backwards. The left half shows a source clip; the clip can be trimmed by scrubbing to the correct frames in the clip in this window and setting in and out points (as in a conventional analogue editing system). The right half of the monitor window is used to display the movie as it has been assembled on the timeline, so that the effects on the whole movie of trims and other edits can be previewed.

Each editor will evolve their own method of working with a particular program, and Premiere, for example, offers alternative means for achieving the same ends. One simple, idealized procedure for editing with Premiere would begin with assembling all the clips for

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 351
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

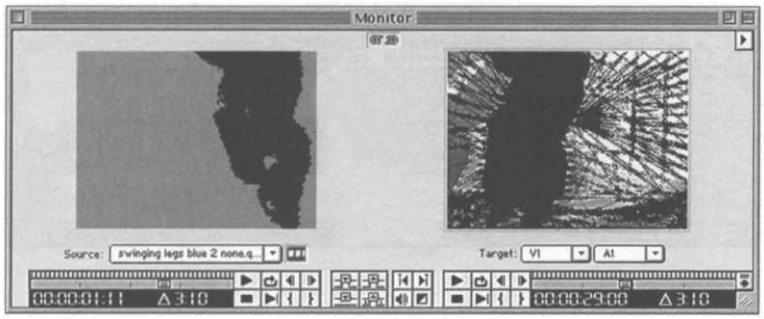


Figure 10.14
Premiere's monitor window

a project — capturing them where necessary, and importing them into the project window, where they are arranged for convenient access. Next, each clip is opened in the monitor window, and roughly trimmed. Note that trimming does not actually discard any frames, as it would with film, it merely suppresses those before the in point and after the out point. If necessary, the in and out points can be readjusted later; if the out point is subsequently moved to a later frame in the clip, or the in point is moved to an earlier one, frames between the old and new points will reappear. The initial trimming operation will leave the editor with the material that they believe should appear in the final movie. The next step is to drag clips from the project window and drop them onto the timeline, assembling them into a rough cut, which can be previewed. Still images can be dragged to the timeline and given a duration; they will behave as clips with no motion. For movies with sound, the picture and sound track can be combined. Almost always, adjustments will have to made, particularly if it is necessary to match up sound and picture. Clips may need to be trimmed again, or more drastic changes may be required, such as the substitution of completely different material when ideas fail to work out.

For some projects, editing will be complete at this stage, but for others, the desired transition from one clip to the next cannot be achieved simply by cutting directly. Using other transitions changes the style, rhythm and mood of a piece. A dissolve, for example, in which one clip fades into another, is less emphatic than a cut, and tends to convey a sense of gradual change or smooth flow from one thing to another. More fanciful transitions, such as wipes, spins and page turns, draw attention to themselves, and function almost as decoration. As most transitions can be described relatively easily in terms of mathematical operations on the two clips involved, digital

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 352
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

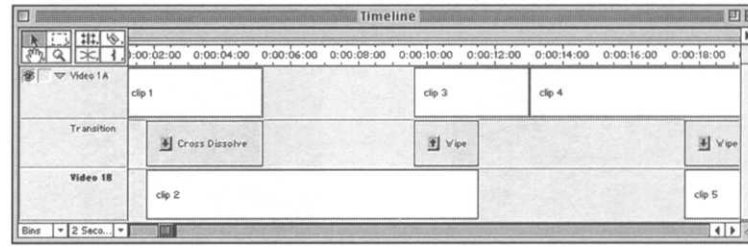


Figure 10.15
Applying transitions in Premiere

video editing software usually offers a vast range of possibilities — Premiere has 75 transitions built in (most of them showy gimmicks), plus a means of defining your own — of which the majority will never be used by a good editor.

Premiere's approach to adding transitions other than cuts is based on the three machine method of editing video. Each transition always involves exactly two clips, conventionally referred to as its A and B clips. These are combined on video track 1, which — uniquely — can be expanded into tracks 1A and 1B, with a special transition track between them. The A and B clips are positioned on tracks 1A and 1B so that they overlap in time; a transition is dragged from the transition palette onto the transition track where A and B overlap (see Figure 10.15). Its duration is automatically adjusted to fill the region of overlap. Many transitions are parameterized: for example, the 'clock wipe' transition may play clockwise or anti-clockwise. Parameters can be set in a dialogue box, brought up by double-clicking the transition on the timeline.

There are two important practical differences between cuts and other transitions. Firstly, in a cut, the two clips are butted; in all other transitions, they overlap, and some part of each contributes to the resulting picture. For sequences which are not going to be edited just with basic cuts it is necessary to ensure that when shot and captured to disk, each clip has sufficient extra frames, beyond the period where it is to play alone, to cover the transition. Secondly, because image processing is required to construct the transitional frames, transitions must be rendered, unlike cuts, which can be implemented simply by copying. Hence, there will inevitably be some loss of image quality where dissolves and so on are used instead of straight cuts.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 353
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Digital Video Post-Production

Video editing is primarily concerned with the arrangement of picture through time, and its synchronization with sound. While frame accuracy may be required to set precise in and out points, or to eliminate faulty footage, for example, it is rarely necessary while editing to consider whole sequences of footage as individual frames. In post-production procedures, however, the temporal dimension may assume less importance, and a movie may be dealt with as a sequence of individual still images. Most digital post-production tasks can be seen as applications of the image manipulation operations we described in Chapter 5 to the images in such a sequence. Two important classes of post-production operation that can be described in these terms are those concerned with image correction, and with compositing.

A video sequence may suffer from the same defects as a single image: for example, it may be over- or under-exposed or out of focus; it may have a colour cast, or it may display unacceptable digitization artefacts. Each of these defects has its characteristic remedy: adjust the levels, sharpen, or apply a Gaussian blur. Post-production systems therefore provide the same set of adjustments as image manipulation programs — some support the use of Photoshop plug-ins — but allow them to be applied to sequences instead of a single image.

Most adjustments have parameters, such as the slider positions for the levels controls. When adjustments are made to sequences, it may be appropriate to use the same parameter values for each image, or it may be preferable for them to change. If, for example, a whole, mostly static, sequence has been shot under incorrect lighting, the same correction will probably be needed for every frame, so the levels can be set for the first, and the adjustment will be applied to as many frames as the user specifies. If, however, the light fades during a sequence, when it was intended to remain constant, it will be necessary to increase the brightness gradually to compensate. It is possible to apply a suitable correction to each frame individually, and this may occassionally be necessary, but often it is adequate to specify parameters at a few key frames and allow their values at intermediate frames to be interpolated. We will show in Chapter 11 how varying parameter values over time can be used to achieve certain special effects.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 354
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Just as some of the image manipulation operations we described in Chapter 5 combined separate images or layers into a composite result, so some post-production operations combine separate video tracks into a composite. In Premiere, as we have seen, video track 1 is special, inasmuch as it can be used for transitions. This gives it a privileged status. All other video tracks can only be used for superimposing picture over video track 1. As with still images, for superimposition to achieve anything useful, some parts of the superimposed tracks must be transparent. In video, selecting transparent areas is called *keying*. Good video editing and post-production software will offer several different keying methods.

The method of blue screening, which we described for single images in Chapter 6, has long been used in video for inserting isolated elements into shots. Traditional examples of its use include adding models to live footage, or placing actors in impossible or dangerous situations. Digital post-production systems support both traditional blue screening, where the actor or model is shot in front of a screen that is a particular shade of blue and then the blue channel is removed, and a more general form of *chroma keying*, where any colour in a scene can be selected and designated as transparent.[23] Chroma keying is essentially the same as building an alpha channel from a selection made using a magic wand tool. An alternative is *luma keying*, where a brightness threshold is used to determine which areas are transparent. Compositing on the basis of luma keying closely resembles the layer blending we demonstrated in Chapter 3.

23
Compare this with the use of one colour from a palette to designate transparency in a GIF image.

As you might expect, it is possible to select a transparent area explicitly, using selection tools to create a mask. In film and video, a mask used for compositing is called a *matte*. Mattes are frequently used for removing unwanted elements, such as microphone booms, from a scene,[24] or for allowing live footage to be combined with a still image. A typical example of such a use of mattes occurs when actors are filmed on a set, containing just a few foreground elements such as trees and plants. The top of the picture (typically just a plain backdrop on the set) is matted out, and subsequently replaced with a painting of a landscape, so that the foreground scene appears to be taking place in front of a mountain range, a medieval castle, or whatever. Mattes can also be used for split-screen effects.

24
In this case, the matte is called a *garbage matte*.

Another way of creating transparency is to use an alpha channel created in some other application. This is often the most satisfactory method of creating mattes to be used in conjunction with

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 355
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

still images, since Photoshop provides much more sophisticated selection tools than Premiere, for example. Alternatively, an imported greyscale image can be used as a matte.

Since a video clip is a sequence of images, a new possibility arises that is not present when single images are being composited: transparency can be made to vary over the course of the sequence (that is, to vary over time when it is played back). This happens automatically with chroma and luma keying, as the colour and brightness distribution changes between frames. In order to produce masking that varies over time, it is necessary to use a sequence of masks as the matte. Such a sequence — often called a *travelling matte* — is naturally stored in a separate video track, when it is called a *track matte*. Although, in principle, the content of any suitable video track could be used as a travelling matte, a specially designed sequence of masks will often be required in practice. Track mattes may be created painstakingly by hand, but are more usually generated from a single still image, by applying simple geometrical transformations over time to create a varying sequence of mattes, in a way that we will describe in Chapter 11. Travelling mattes are often used in title sequences, where they are especially favoured for news bulletins.

All of the post-production facilities we have described can be found in Premiere and other desktop video editing applications, but only in a relatively crude form. For serious post-production, dedicated applications such as Adobe After Effects are preferred. Although these do not offer any radically different operations, they do provide much greater control over their application, and implement them more accurately. Where an editing program may allow you to apply a filter to a clip, and vary it over time, a post-production program will provide a wide range of controls for the filter's parameters, and for the more sophisticated interpolation between key frames that it offers. For example, in Premiere, the parameter values are interpolated linearly between key frames, whereas in After Effects, the interpolation can also use Bézier curves, which can be specified using handles and control points, as they can in Illustrator.

⇨ Another post-production task that is perhaps less important for multimedia production than for commercial film and video is titling. Editing programs' titling facilities are usually barely adequate, simply allowing the superposition of text over video — Premiere did not provide any means of creating rolling credits until version 5.0. Sophisticated title and credit sequences are often best made by

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 356
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

using graphics applications to create the text and graphic elements, which are then animated using a post-production tool such as After Effects, in a way that will be described in Chapter 11.

Preparing Video for Multimedia Delivery

Editing and post-production are performed in roughly the same way whether the final video is intended for multimedia delivery — off-line or on-line — or for conventional transmission or recording on video tape. For multimedia, an additional step of preparing the material for delivery is usually required. The reason for this step is the need to cope with limitations of the final delivery medium and playback platforms that are not usually relevant on the platform used for editing and post-production. (Recall from Chapter 2 that multimedia production usually takes place on specially equipped top-of-the-range desktop machines or workstations, whose performance far exceeds that of consumer machines.) Compromises must be made at this stage, which will involve choosing what is to be sacrificed in order to bring the resource requirements of video within the capabilities of delivery media and low-end machines. Different material will permit and suggest different choices.

What might be sacrificed in this way? The possibilities include frame size, frame rate, colour depth, and image quality. Reducing the size of video frames is usually a relatively painless way of reducing the size and bandwidth of video files. People usually sit close to their monitors, so a large picture is not necessary, and monitor resolution is usually better than that of television sets, so a down-sampled image on a computer screen can appear smoother than the images we are familiar with on television (assuming the down-sampling is done intelligently). Similarly, reducing the frame rate is often acceptable: the illusion of continuous motion can often be satisfactorily maintained with frame rates around 12 fps. Higher frame rates are needed to eliminate flicker only if the display is refreshed at the frame rate. This is not the case with computer monitors, which are refreshed at a much higher rate from VRAM. By using quarter frames and reducing the frame rate from 30 fps to 15, the volume of data is reduced by a factor of eight.

A further factor of three can be obtained by reducing the colour depth from the 24 bits usually used for video to eight bits. As we explained in Chapter 6, this can be done by limiting the colours

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 357
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

to those in a standard palette (usually the Web-safe palette), using indexed colour with a custom palette, or by reducing to 256 shades of grey. We argued in Chapter 6 that the last option may be preferable to using a limited range of colours. Unfortunately, not all video codecs support the use of greyscale images. Among those that do not are Sorenson, Intel Indeo, and H.263[25], all of which are otherwise well suited to multimedia.

25
See page 344.

If these compromises are unacceptable, or — as may easily be the case — they do not sufficiently reduce the resource requirements of your video, it will be necessary to squeeze bytes out with compression, at the cost of a loss of image quality. We have already described suitable codecs, and their characteristics. It should be emphasized that, despite all the clever algorithms they use and despite the claims that are sometimes made, all codecs introduce visible artefacts when very high compression ratios are needed. It may be necessary to apply filters in order to minimize their effects, but this will also result in a further loss of picture information.

As well as taking steps to minimize the size of video, it is also necessary to ensure that it is fit to play on any platform. QuickTime movies prepared on a Macintosh computer must be flattened, for example, so that they do not store any data in the resource fork that is unique to MacOS files. Where necessary, movies must be made self-contained. That is, where the file format allows pointers to other files to be used as part of a movie, these pointers must be replaced by the data they refer to. It may be necessary to produce different versions of a movie for different platforms, to compensate for the different gamma of PC and Mac monitors, for example.[26] For video that is to be delivered over a network it is common practice to produce a range of different versions matched to the speed of users' network connections. QuickTime allows different versions to be combined into a single movie.

26
It may sometimes be necessary to produce platform-specific versions at the editing stage, too. For example, if a movie contains product advertisements, there is little point in including advertisements for products that only run under Windows as part of the Macintosh version.

⇨ The processing described in this section is not always necessary even for video that is being incorporated into a multimedia production. If the production is to be presented in a kiosk or as part of an exhibition, where the producer determines what hardware is to be used, the video can be delivered at whatever quality the hardware supports. If a suitable video card is known to be available, for example, the codec that was used for capture can be used for delivery; if terabytes of storage can be supplied, uncompressed video can be used, and so on. Cross-platform issues are irrelevant in such situations.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 358
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Video editing programs provide facilities for performing the necessary preparation for delivery, but as with post-production tasks, their facilities are relatively crude, and it is better to use a dedicated application, such as Media Cleaner Pro. This provides improved control over settings for codecs, and integrates compression with the other tasks we have described for preparation. Suitable filters can be applied to mitigate the effects of aggressive compression, and multiple versions can be constructed at the same time, and integrated. Additional facilities that are especially useful at this stage of production include a data rate analyzer, and a split-pane preview window for showing the effects of different settings.

Streamed Video and Video Conferencing

It's one thing to play a pre-recorded video clip from a hard disk, DVD or even a CD-ROM; it's quite another to play video over a network. By this we mean delivering a video data stream from a remote server, to be displayed *as it arrives*, as against downloading an entire video clip to disk and playing it from there. Such *streamed video* resembles broadcast television, in that the source video is held on the server, which acts like a TV transmitter sending out the signal, which is played back straight away on a client machine. Downloading the entire clip is as though the TV company sent a courier round with a videotape whenever you wanted to watch a programme. Streamed video opens up the possibility of delivering live video, bringing one of the modes of conventional broadcasting to video on computers. It goes beyond conventional broadcast TV in this area, though, because it is not restricted to a single transmitter broadcasting to many consumers: any suitably equipped computer can act both as receiver and transmitter, so users on several machines can communicate visually, taking part in what is usually called a *video conference*.

The fundamental obstacle to streamed video is bandwidth. As we showed earlier, even the heavily compressed and down-sampled quarter-frames of SIF MPEG-1 video require a bandwidth of 1.86 Mbits per second. For now, therefore, decent quality streamed video is restricted to local area networks and T1 lines, ADSL and cable modems; dial-up Internet connections using V90 modems or basic rate ISDN (never mind slower modems) cannot handle the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 359
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

required data rate. Even where the bandwidth is available, the network has to be capable of delivering data with the minimum of delay, and without undue 'jitter' — a variation in the delay that can cause independently delivered video and audio streams to lose synchronization. We will return to this subject in Chapter 15.

⇨ It is likely that a large proportion of the population will be restricted to lower speed connections for some time to come. In the domestic sector, video is most likely to be streamed over broadcast digital television; video conferencing is unlikely to be an option, and the shape of interactivity over the digital TV networks will depend on market forces and politics at least as much as on the technological capabilities of the delivery medium.

This is not to say that streaming video over slow dial-up Internet connections is impossible, just that the quality is poor.

It may help you to understand the nature of what we will sometimes call *true streaming* by contrasting it with alternative methods of video delivery you may meet on the World Wide Web. The simplest method is embedded video, where a movie file is transferred from a server to the user's machine, where it is played back from disk once the entire file has arrived. A refinement of this method is called *progressive download* or *HTTP streaming*. With this mode of delivery, the file is still transferred to the user's disk, but it starts playing as soon as enough of it has arrived. This will be when the time it will take for the remainder to be downloaded is equal to the duration of the entire movie. This is illustrated in Figure 10.16. There is usually an appreciable delay before playback starts, since progressively downloaded movies are typically made with a data rate that exceeds the bandwidth of the network connection, in order to maintain quality. The movie file usually remains on the user's hard disk — at least in their Web browser's cache — after playback is completed. Thus, enough disk space to store the whole movie must be available, so progressive download cannot be used for huge files, such as complete feature films. Also, since an entire file is downloaded, this method of delivery cannot be used for live video, nor does it allow you to skip over parts of the file without downloading them.

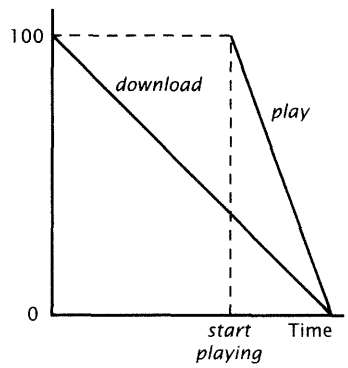


Figure 10.16
Progressive download

In contrast, true streaming video is never stored on the user's disk. A small buffer may be used to smooth out jitter, but effectively each frame in the stream is played as soon as it arrives over the network. This means that streams can be open-ended, so true streaming can be used for live video, and the length of a recorded movie

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 360
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

that is streamed is limited only by the amount of storage available at the server, not by the viewer's machine. Random access to specific points in a stream is possible, except for live streams. True streaming is thus suitable for 'video on demand' applications.[27] Its drawback is that the network must be able to deliver the data stream fast enough for playback. Looked at from the other side, this means that the movie's data rate, and thus its quality, is restricted to what the network can deliver. Even with the best connections, Internet streamed video will not provide the picture quality of broadcast television.

27
The fact that no copy of the entire movie is held on the user's machine makes it even more attractive where copyright material is involved.

⇨ Why bother, then? The answer is sometimes summed up in the slogan, 'Every Web site a TV station'. Streaming video server software is available free, or (for a small number of simultaneous streams) at a modest cost. Anyone with a permanent Internet connection can therefore transmit streamed video content; many ISPs provide streaming facilities to their customers who do not run their own servers. Internet streaming is thus much more accessible than conventional TV broadcasting facilities, and so it provides a new communication channel for minority voices and unconventional video (experimental and underground work, as well as the more notorious sort) that is denied accesss to conventional channels. As we will explain in Chapter 15, though, the broadcasting establishment may not be undermined to the extent that might seem possible... or desirable.

Another answer, more often given by conventional broadcasters, is that streamed video can be interactive. It is not entirely clear, though, what sort of interaction, if any, people want to have with video. Facilities that have been proposed for broadcast digital TV appear to be restricted to allowing viewers to select between different camera angles, and to call up 'instant replays' on demand. Initial provision of such facilities, for example by the satellite TV company BSkyB, is being targeted at sports coverage. It is not clear that this sort of interactivity, when provided over the Internet, would compensate for poor picture quality.

The two factors of democratic access and interactivity are combined in video conferencing, which may currently be the most successful application of streamed video, but is somewhat marginal from our viewpoint on multimedia, opening up, as it does, questions better considered in the context of computer-supported collaborative working.

28
True streaming was added to QuickTime with version 4.0.

The two leading architectures for streaming video over the Internet are *Streaming QuickTime* ,[28] and Real Networks' *RealVideo* . They

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 361
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

have several features in common. Both are based on open Internet standard protocols, particularly RTSP, the Real Time Streaming Protocol, which is used to control the playback of video streams, which are carried over the network using the Real Time Protocol, RTP. These protocols are described in Chapter 15. RealVideo and Streaming QuickTime streams can both be embedded in Web pages; Streaming QuickTime streams can also be embedded in any application that incorporates QuickTime. Both architectures provide a means of providing several different versions of a movie compressed to match the requirements of different types of connections — a 28.8 kbps version, a 56 kbps version, a T1 version, and a cable modem version may all be provided so that, to the user, there only appears to be one movie; the server chooses the appropriate one to fit the speed of the user's connection. Tools are available for integrating servers for either format with capture facilities, allowing live video to be streamed to the Internet. As well as true streaming, progressive download is supported by both QuickTime and RealVideo. There is really little to choose between the technologies, but, since RealVideo has been around for longer, the production tools and server are more mature, and it is more likely that users will have the necessary plug-in and player to view it. In Streaming QuickTime's favour is the fact that it is just a variety of QuickTime, which can be embedded in any application, not just in a Web browser or its own special-purpose player.

Codecs for Streamed Video and Video Conferencing

Since Streaming QuickTime is essentially just QuickTime delivered in a special way, all of the QuickTime codecs can be used for streaming. However, to obtain sufficient compression to stream a movie over most network connections, these have to be used on their lowest quality settings, and the movie will generally have to be scaled down to a small frame size. Better results may be obtained using codecs designed specifically for video conferencing, which are designed for streaming video at low bit rates.

H.261 (named after the ITU-T Recommendation that defines it) was designed for video conferencing over ISDN. Since each ISDN data channel has a bandwidth of 64 kbps, H.261 was defined to operate at data rates that are an exact multiple of this rate. Hence, it is sometimes referred to as $p \times 64$, where, as p ranges from 1 to 30 (the maximum number of ISDN channels), the frame rate and the quality

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 362
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

of the video stream scale up. An important feature of H.261 is that the recommendation specifies a maximum delay of 150 milliseconds due to the codec. This is because a video conference cannot tolerate longer delays without becoming disjointed — participants require immediate visual feedback to preserve the illusion of presence. Another singular feature is that the frame size is restricted to one of two values: *Common Intermediate Format (CIF)* is 352×288 pixels;[29] at lower data rates, a frame size of 176 × 144 pixels — i.e. one quarter that of CIF, hence *QCIF* — is used. Generally, if $p < 6$, that is, the available bandwidth is less than 384 kbps, QCIF must be used. There is a non-standard 'sub-QCIF' format of 128 × 96 for very low bandwidth applications. Whichever size is used, the video signal is chroma sub-sampled to 4:2:0.

29
Depending on how you measure your active samples.

H.261 uses a DCT-based compression algorithm, with motion compensation. It can be seen as a precursor of MPEG-1, but it does not use anything like B-pictures. Based on the additional experience gained from MPEG, a successor to H.261, *H.263* has been developed. This is very similar to MPEG, but it goes further by providing a means of combining a P-picture and a B-picture into a single *PB-frame*. H.263 was targeted at very low bit rates indeed: its primary target rate is about 27 kbps — the speed of a V.34 modem — and its specification is for bit rates less than 64 kbps. It achieves these rates by using all the compression techniques of MPEG — 4:2:0 chroma sub-sampling, DCT compression with quantization, run-length and variable-length encoding of coefficients, and motion compensation with forward and backward prediction — to compress a QCIF picture at frame rates as low as 3.5 frames per second. It does produce recognizable moving pictures at sufficiently low rates to be used for streaming and conferencing over dial-up Internet connections, but the quality is best described as 'better than nothing'.

For streaming over faster connections, the Sorenson codec is recommended for Streaming QuickTime. RealVideo uses its own proprietary codec.

Further Information

[Poy96] provides an expert description of digital video technology. [Oha93] and [Tho93] describe different aspects of film and video

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 363
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

editing; parts of [Kaw92] are relevant, too. [Tud95] is a good short introduction to MPEG compression. [VD96] is an entertaining account of the use of digital techniques in high-budget film production. We will return to the technology underlying streaming video in Chapter 15.

Exercises

1. Under what circumstances will you need a hardware codec when working with video?

2. Specify the ideal video components of a system for the following projects:

   (a) A video interview with a celebrity, to be recorded at their home, and incorporated into a multimedia production on CD ROM for which you will need to extract high quality still images as well as the video footage.

   (b) A live video conference to be conducted across the Internet.

   (c) A video recording of the close finish of a 100 metre sprint, for a multimedia production on the Olympic games, which will allow the user to determine the winner for themselves.

   (d) A surveillance system in a bank which would provide visual evidence for a forensic multimedia presentation in a court of law.

   (e) Video footage of a fast flowing mountain stream, which is to be presented as smoothly as possible in slow motion, in a small sized window on a web page.

   (f) Time-lapse footage of the opening of a flower, for use on a Web site devoted to gardening.

3. Specify a practical system, based only on the equipment available to you, for realising the same projects.

4. Suppose you were involved in the design of a multimedia software application for domestic use, intended to allow users to create 'home' multimedia productions such as a record of a child's birthday to send on CD ROM to a grandparent

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 364
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

on the other side of the world. What assumptions will you make about video for this program, and what facilities will you supply?

5. How much storage would be occupied by a 90 minute feature film stored in each of the following forms?

   (a) CCIR 601.

   (b) MP@ML MPEG-2.

   (c) DV.

   (d) MPEG-1 in SIF.

   For each format, indicate what applications it is suitable for.

6. What effect will each of the following common video idioms have on a compression scheme that includes temporal compression?

   (a) Cuts;

   (b) Dissolves;

   (c) Hand-held camera work;

   (d) Zooms;

   (e) Pans.

   In which cases does motion compensation help?

7. Suppose an MPEG encoder uses the nine frame sequence IBBPBBPBB as a GOP. Draw a diagram showing the dependencies between the first eighteen frames of a compressed clip produced by this encoder. Show how the pictures would be reordered into bitstream order. Explain carefully why the pattern of I, P and B pictures in the bitstream order of the first nine frames is different from that of the second nine frames.

8. Capture a short clip[30] from TV or a video tape of each of the following:

   (a) A classic black and white *film noir.*

   (b) A contemporary soap opera.

   (c) A silent comic piece by Chaplin or Keaton.

   (d) A serious documentary.

   (e) A musical comedy in Glorious Technicolor.

30 Pay attention to copyright and use these clips only for your own work. Do not attempt to publish or broadcast them in any medium, including from a Web site.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 365
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Observe how each has been edited to give it its particular character.

Re-edit each clip to give it the style and feel of one of the others. For example, make a silent comedy clip out of the documentary, or *film noir* clip out of the musical comedy.

(a) Do this simply with basic transitions, i.e. cuts, dissolves, and wipes.

(b) If post-production tools are available to you, take this further, by changing colours, contrast, or the quality of sound. For example, render the contemporary soap opera in grainy black and white with crackly sound, or the re-edited *film noir* in three-colour Technicolor.

9. A user starts to download a 905 kilobyte movie of 30 seconds duration, using progressive download over a connection that provides an average data rate of 2500 bytes per second. How long will it be before the movie starts to play? Why might the user experience jerky playback?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 366
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

11 Animation



Animation may be defined as the creation of moving pictures one frame at a time; the word is also used to mean the sequences produced in this way. Throughout the twentieth century, animation has been used for entertainment, advertising, instruction, art and propaganda on film, and latterly on video; it is now also widely employed on the World Wide Web and in multimedia presentations.

To see how animation works, consider making a sequence of drawings or paintings on paper, in which those elements or characters intended to change or move during the sequence are altered or repositioned in each drawing. The changes between one drawing and the next may be very subtle, or much more noticeable. Once the drawings are complete, the sequence of drawings is photographed in the correct order, using a specially adapted movie camera that can advance the film a single frame at a time. When the film is played back, this sequence of still images is perceived in just the same way as the sequence of frames exposed when live action has been filmed in real time: persistence of vision causes the succession of still images to be perceived as a continuous moving image. If you wish to convey the illusion of fast movement or change, the differences between successive images in the sequence must be much greater than if the change is to be gradual, or the movement slow.

Etymologically, 'animate' means 'to bring to life', which captures the essence of the process: when played back at normal film or video

. DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 367
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

speeds, the still characters, objects, abstract shapes, or whatever, that have been photographed in sequence, appear to come to life.

As film is projected at 24 frames per second, drawn animation, as we have just described it, technically requires 24 drawings for each second of film, that is, 1440 drawings for every minute — and even more for animation made on video. In practice, animation that does not require seamlessly smooth movement can be shot 'on 2s', which means that two frames of each drawing, or whatever, are captured rather than just one. This gives an effective frame rate of 12 frames per second for film, or 15 for NTSC video.

If an animation is made solely from drawings or paintings on paper, every aspect of the image has to be repeated for every single frame that is shot. In an effort to reduce the enormous amount of labour this process involves, as well as in a continuing search for new expressive possibilities, many other techniques of animation have been devised. The most well known and widely used — at least until very recently — has been *cel animation.* In this method of working, those elements in a scene that might move — Homer Simpson, for example — are drawn on sheets of transparent material known as 'cel', and laid over a background — the Simpsons' living room, perhaps — drawn separately. In producing a sequence, only the moving elements on the cel need to be redrawn for each frame; the fixed part of the scene need only be made once. Many cels might be overlaid together, with changes being made to different ones between different frames to achieve a greater complexity in the scene. To take the approach further, the background can be drawn on a long sheet, extending well beyond the bounds of a single frame, and moved between shots behind the cels, to produce an effect of travelling through a scene. The concepts and techniques of traditional cel animation have proved particularly suitable for transfer to the digital realm (see Figure 11.1).

Largely because of the huge influence of the Walt Disney studios, where cel animation was refined to a high degree, with the use of multi-plane set-ups that added a sense of three-dimensionality to the work, cel has dominated the popular perception of animation. It was used in nearly all the major cartoon series, from *Popeye* to the *Simpsons* and beyond, as well as in many full-length feature films, starting with *Snow White and the Seven Dwarfs* in 1937. However, from the very beginnings of moving pictures in the 1890s, animation has been successfully created by employing a variety of other means. Many artists do indeed work by drawing each frame



Figure 11.1
A cel-like digital animation

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 368
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043











separately on paper, while others, even more painstaking, have painted directly on to film, or scratched the emulsion of blackened film stock; others have worked with sand or oil paint on glass, or chalks on paper or card, making changes to the created image between every shot; still others have manipulated front or back lit cut-outs under the camera — Terry Gilliam's very strange work for the *Monty Python* TV series is a well-known example of cut-out animation. Sometimes animators have invented a completely new way of working for themselves, such as Alexeieff and Parker's pin screen, in which closely spaced pins are selectively pushed through a board and lit so that the shadows they cast form an image, which is changed between each shot.

A distinct alternative to all of these essentially two-dimensional forms is three-dimensional, or *stop-motion* animation. This encompasses several techniques, but all use miniature three-dimensional sets, like stage sets, on which objects are moved carefully between shots. The objects may include articulated figures, whose limbs can be repositioned, or solid figures whose parts are replaced, or substituted, between shots, to produce an effect of gestures, walking, and so on. Figures and other objects made out of a malleable modelling material, such as Plasticine, may be used instead; these can be manipulated between shots, to produce both natural movement, and otherwise impossible changes and transformations. This latter form of animation — often called *clay animation* — has achieved recent prominence with the work of the Aardman studios whose output includes the *Wallace and Gromit* films.

Although it may be convenient to consider the various techniques of animation separately, hybrid forms of animation are often produced — mixing cel and 3-D, for example. There is also a long tradition of combining animation with live footage. The most celebrated example of this is perhaps *Who Framed Roger Rabbit?* (1988), but a mixture of live action and animation was employed in some of the earliest films ever made, including Georges Méliès well known 'trick films', and Max Fleischer's *Out of the Inkwell* series of the 1920s, which did much to popularize animation as a form of entertainment. Recently, the eager adoption of digital technology by the film industry has led to a substantially increased use of animation in conjunction with live action, particularly in special effects movies, such as *Jurassic Park*, that have provided some of the major box-office hits of the 1990s.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 369
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

⇨ It is perhaps not always realised by an audience that much of what they perceive as 'special effects' has been achieved by basic animation techniques, whether traditional, as in the 1933 classic *King Kong* and many other monster movies, or digital, as in the TV series *Babylon 5*, for example.

All of the established forms of animation have their counterparts in the digital realm. Moreover, digital technology affords new opportunities for using animation and techniques derived from it in new contexts.

Captured Animation and Image Sequences

As we will see, digital technology has brought new ways of creating animation, but computers can also be used effectively in conjunction with the older methods discussed above, to produce animation in a digital form, suitable for incorporation in multimedia productions. Currently, preparing animation in this way — using digital technology together with a video camera and traditional animation methods — offers much richer expressive possibilities to the animator working in digital media than the purely computer-generated methods we will describe later in this chapter.

Instead of recording your animation on film or videotape, a video camera (either a digital camera or an analogue camera connected through a video capture card) is connected directly to a computer, to capture each frame of animation to disk — whether it is drawn on paper or cel, constructed on a 3-D set, or made using any other technique that does not depend on actually marking the film. Instead of storing the entire data stream arriving from the camera, as you would if you were capturing live video, you only store the digital version of a single frame each time you have set up a shot correctly. Most digital video editing applications provide a facility for *frame grabbing* of this sort. Premiere, for example, offers a `Stop Frame` command on its `Capture` menu. This causes a recording window to be displayed, showing the current view through the camera. You can use this to check the shot, then press a key to capture one frame, either to a still image file, or to be appended to an AVI or QuickTime movie sequence. You then change your drawing, alter the position of your models, or whatever, and take another shot. Frames that are unsatisfactory can be deleted; an

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 370
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

option allows you to see a ghost image of the previously captured frame, to help with alignment and making the appropriate changes. When you have captured a set of frames that forms a sequence, you can save it as a QuickTime movie or a set of sequentially numbered image files (see below). The latter option is useful if you want to manipulate individual images in Photoshop, for example.

For certain types of traditional animation, it is not even necessary to use a camera. If you have made a series of drawings or paintings on paper, you can use a scanner to produce a set of image files from them. You can also manipulate cut-outs on the bed of a scanner, almost as easily as under a camera. A film scanner will even allow you to digitize animation made directly onto film stock. You might be able to use a digital stills camera instead of a video camera, provided it allows you to download images directly to disk. In all of these cases you are able to work at higher resolution, and with a larger colour gamut, than is possible with a video camera.

For drawn or painted animation you can dispense with the external form and the digitization process entirely by using a graphics program to make your artwork, and save your work as a movie or as a sequence of image files.

⇨ Sequences of image files provide a very flexible representation of an animation. Individual files can be opened in a graphics program to be altered; single files can be removed from the sequence, replaced or added. The sequence can then be imported into a video editing application and converted into an AVI or QuickTime movie. However, managing a collection of image files can become complicated, especially if you eventually want to import them into a video editing program. In order for this to be possible without tying you to a particular combination of programs, the files' names must conform to some convention. For example, on the Macintosh, Premiere can only import a sequence of PICT files if they are all in the same folder, and all the files have a suffix consisting of a period followed by the same number of digits, for example `Animation.001`, `Animation.002`, ...`Animation.449`. (Failure to provide the necessary leading zeroes will have consequences that you can probably guess at.) If you make any changes to the set of images, you must take care not to disturb the numbering, or to adjust it if necessary.

Painter provides special support for animators wishing to draw or paint animations digitally,[1] in the form of features that resemble some of the ways in which traditional animators work with a stack

1 Recall from Chapter 3 that Painter provides tools that simulate natural media, so you can make pictures that look (somewhat) as if they were made with conventional art materials.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 371
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

of drawings. A Painter *frame stack* is a set of images, all of the same dimensions and colour depth. When you create or open a frame stack, you are provided with controls for navigating through it — advancing or retreating one frame, or going to the beginning or end of the stack — and playing it by having each frame in the stack displayed in turn. You create an animated sequence by painting on each of the frames in the stack. By turning on *onion skinning* you can make up to four frames adjacent to the one you are currently working on visible as ghost images, to help you line up the static parts of the animation and judge the amount by which you need to change the moving elements. This practice is modelled on the way animators sometimes use a light box to help construct a series of frames. By working on thin paper (hence the name 'onion skin') on a surface illuminated from below, they can lay the sheet they are working on over the one preceding or following it, which will show through the top sheet as a ghost image in just the same way.

Plate 20 is a screenshot of a Painter framestack being worked on, showing the navigational controls and the ghost images of the nearby frames. Note also the thumbnails of these frames.

Because it is a computer program, Painter (and similar applications) can help the animator in ways that are difficult or impossible when working with physical media. The most characteristically computerized facility is the ability to record a sequence of operations as a *script* that can subsequently be played back to apply the same operations to a new image. For example, you could record everything you did while drawing a static element in the first frame of a stack, and then, when you came to work on the next frame, simply play back the recording to redraw that element automatically, leaving you with just the changing parts of the animation to do by hand. You can also use a script to record the application of a combination of effects, such as adjustments to brightness and contrast, or colour balance, to one image, and then have the script applied to an entire frame stack to alter every frame in the same way.

Alternatively, a framestack may be used to create animation by progressively altering one single image, simulating traditional methods of working in paint or sand on glass, etc. Painter is especially appropriate for this, given its extensive and customisable range of natural media and brushes. In this case each new frame starts as a copy of the preceding frame, which is then partly erased and repainted to create the animation. (Painter allows you to add

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 372
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

frames on to a framestack one at a time, with an option to make each new frame added a copy of the last in the stack at that time.)

As well as letting you save a frame stack as a sequence of image files, Painter will let you save it as a QuickTime or AVI movie. It will also let you open a movie, transforming it into a frame stack when you do so. This offers new possibilities. You can, for example, paint onto or otherwise alter original video material, which is one way of adding animation to live action.[2] Another option is to trace, frame by frame on a new framestack, selected elements from a live action video clip opened as a separate framestack. (Painter provides special facilities for doing this). This process, whether achieved digitally or by older means, is what is properly referred to as rotoscoping, and has long been used to create animation that accurately reproduces the forms and natural movements of people and animals.

2
In computer graphics circles, this process of painting on to existing video frames is sometimes called 'rotoscoping', but the use of the term is inaccurate, as explained below.

⇨ Rotoscoping is named after the *rotoscope*, a device patented by Max Fleischer (of *Betty Boop* and original animated *Popeye* fame) in 1915. Fleischer's device projected movie footage, one frame at a time, onto a light table, giving a back projected still image over which the animator could place a sheet of animation paper. When the tracing of one frame was complete, the film was advanced to the next by means of a hand crank.

Instead of using a set of still image files to hold an animation sequence, you can sometimes use a single 'image' file to hold several images. While a surprising number of file formats — including, but not exclusively, formats intended for use with animation software — offer this facility, by far the most common is GIF.

GIF files' ability to store a sequence of images has been used to provide a cheap and cheerful form of animation for Web pages. Most Web browsers will display each image contained in a GIF file in turn when they load the file. If the displaying happens fast enough, the images will be seen as an animation. The GIF89a version of the format provides for some optional data items that control the behaviour of an *animated GIF*, as these files are called. In particular, a flag can be set to cause the animation to loop, either for a stipulated number of times or indefinitely, and a minimum delay between frames, and hence a frame rate, can be specified. However, animated GIFs do not provide a very reliable way of adding animated features to Web pages. As with most aspects of a browser's behaviour, the way in which animated GIFs are displayed can be changed by users — looping can be turned off, animation can be

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 373
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

prevented, and if image loading is disabled, animated GIFs will not appear at all — and not all browsers offer a proper implementation. The main advantage of animated GIFs is that they do not rely on any plug-in, or the use of scripting (see Chapter 14).

Several free or inexpensive utilities are available on the major platforms for combining a set of images into a single animated GIF; Painter can export a frame stack as an animated GIF, and Premiere and Flash allow you to save a movie in this form, too. Potentially, therefore, GIF files can be used to store any form of animation. However, even when GIF animation is properly implemented and enabled, it has many shortcomings. You cannot add sound; you are restricted to a 256 colour palette; your images are losslessly compressed, which may conserve their quality, but does not provide much compression, a serious consideration that effectively prevents the use of this format for any extended animation sequences. Usually, each frame of an animated GIF is displayed by the browser as it arrives. Network speeds mean that there may be excessive, and probably irregular, delays between frames, making any frame rate that may be specified in the file irrelevant. However, if an animation is set to loop, once it has played through the first time it will have been copied into the browser's local cache (unless it is too big), and subsequent loops will play at a speed only limited by the user's processor and disk (which are completely unknown to the animator). In general, there is little chance of an animated GIF consistently playing back at a sufficiently high frame rate to give smooth animation, unless it is small (and even then, not on older computers). Usually, therefore, animated GIFs are not used for real animation, but for more stylized changing images, often resembling neon advertising signs. Possibly for this reason, by association of ideas, Web page advertising is what animated GIFs are most often used for. It is probably fair to say that, because of the ease with which animated advertisements can be incorporated into a Web page by almost anybody, they have been used for some of the worst animation ever produced.

For animation of any duration, especially if it is accompanied by sound, the best results will be achieved using a video format, and QuickTime has become the standard. Once you have captured or painted an animation sequence and saved it as, or converted it to, QuickTime in the manner previously described, what you have is just an ordinary QuickTime movie, so it can be edited, combined with other clips, have effects applied, and be prepared

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 374
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

for inclusion in a multimedia production, just like any other video clip. However, animation clips may have some distinctive features which affect the way you deal with them. In particular, certain styles of drawn animation tend to feature simplified shapes and areas of flat colour. (This is *not* invariably the case, the characteristics of the images depend on the individual animator's style.) Material of this type may be more amenable to lossless compression than other types of video. QuickTime's *Animation* codec is designed to take advantage of the characteristics of simple cartoon-style drawn animation, which, as we will see later in this chapter, are often shared by computer-generated 3-D animation. Compression is based on run-length encoding (RLE), and, when the codec is used at its highest quality setting, is lossless. There is also a lossy mode that can be used to achieve higher compression ratios. Because it is based on RLE, this codec can compress areas of flat colour well, which is what makes it suitable for animation in the particular styles just mentioned. It works best on images made in the computer: the signal noise that may be introduced by capturing from a video camera can interfere with its effectiveness.

'Digital Cel' and Sprite Animation



Our earlier description of cel animation may have put you in mind of layers, as described in Chapter 3. Layers allow you to create separate parts of a still image — for example, a person and the background of a scene they are walking through — so that each can be altered or moved independently. The frames of an animated sequence can be made by combining a background layer, which remains static, with one of more animation layers, in which any changes that take place between frames are made. Thus, to create an animation, you begin by creating the background layer in the image for the first frame. Next, on separate layers, you create the elements that will move; you may want to use additional static layers in between these moving layers if you need to create an illusion of depth. After saving the first frame, you begin the next by pasting the background layer from the first; then, you add the other layers, incorporating the changes that are needed for your animation. In this way, you do not need to recreate the static elements of each frame, not even using a script.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 375
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Where the motion in an animation is simple, it may only be necessary to reposition or transform the images on some of the layers. To take a simple example, suppose we wish to animate the movement of a planet across a background of stars. The first frame could consist of a background layer containing the star field, and a foreground layer with an image of our planet. To create the next frame, we would copy these two layers, and then, using the move tool, displace the planet's image a small amount. By continuing in this way, we could produce a sequence in which the planet moved across the background. (If we did not want the planet to move in a straight line, it would be necessary to rotate the image as well as displace it, to keep it tangential to the motion path.) Simple motion of this sort is ripe for automation, and we will see in a later section how After Effects can be used to animate Photoshop layers semi-automatically.

Using layers as the digital equivalent of cel saves the animator time, but, as we have described it, it does not affect the way in which the completed animation is stored: each frame is saved as an image file, and the sequence will later be transformed into a QuickTime movie, an animated GIF, or any other conventional representation. Yet there is clearly a great deal of redundancy in a sequence whose frames are all built out of the same set of elements. Possibly, when the sequence comes to be compressed, the redundant information will be squeezed out, but compressing after the event is unlikely to be as successful as storing the sequence in a form that exploits its redundancy in the first place. In general terms, this would mean storing a single copy of all the static layers and all the objects (that is, the non-transparent parts) on the other layers, together with a description of how the moving elements are transformed between frames.

This form of animation, based on moving objects, is called *sprite animation*, with the objects being referred to as *sprites*. Slightly more sophisticated motion can be achieved by associating a set of images, sometimes called *faces*, with each sprite. This would be suitable to create a 'walk cycle' for a humanoid character, for example (see Figure 11.2). By advancing the position of the sprite and cycling through the faces, the character can be made to walk.

QuickTime supports sprite tracks, which store an animation in the form of a 'key frame sample' followed by some 'override samples'. The key frame sample contains the images for all the faces of all the sprites used in this animation, and values for the spatial properties



Figure 11.2
Sprite faces for a walk cycle

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 376
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043



(position, orientation, visibility, and so on) of each sprite, as well as an indication of which face is to be displayed. The override samples contain no image data, only new values for the properties of any sprites that have changed in any way. They can therefore be very small. QuickTime sprite tracks can be combined with ordinary video and sound tracks in a movie.

We have described sprite animation as a way of storing an animated sequence, but it is often used in a different way. Instead of storing the changes to the properties of the sprites, the changed values can be generated dynamically by a program. Simple motion sequences that can be described algorithmically can be held in an even more compact form, therefore, but, more interestingly, the computation of sprite properties can be made to depend upon external events, such as mouse movements and other user input. In other words, the movement and appearance of animated objects can be controlled by the user. This way of using sprites has been extensively used in two-dimensional computer games, but it can also be used to provide a dynamic form of interaction in other contexts, for example, simulations. We will see something similar in Chapter 14 when we consider animating text on Web pages.

Key Frame Animation

During the 1930s and 1940s, the large American cartoon producers, led by Walt Disney, developed a mass production approach to animation. Central to this development was division of labour. Just as Henry Ford's assembly line approach to manufacturing motor cars relied on breaking down complex tasks into small repetitive sub-tasks that could be carried out by relatively unskilled workers, so Disney's approach to manufacturing dwarfs relied on breaking down the production of a sequence of drawings into sub-tasks, some of which, at least, could be performed by relatively unskilled staff. Disney was less successful at de-skilling animation than Ford was at de-skilling manufacture — character design, concept art, storyboards, tests, and some of the animation, always had to be done by experienced and talented artists. But when it came to the production of the final cels for a film, the rôle of trained animators was largely confined to the creation of *key frames*.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 377
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

We have met this expression already, in the context of video compression and also in connection with QuickTime sprite tracks. There, key frames were those which were stored in their entirety, while the frames in between them were stored as differences only. In animation, the meaning has a slightly different twist: key frames are typically drawn by a 'chief animator' to provide the pose and detailed characteristics of characters[3] at important points in the animation. Usually, key frames occur at the extremes of a movement — the beginning and end of a walk, the top and bottom of a fall, and so on — which determine more or less entirely what happens in between, but they may be used for any point which marks a significant change. The intermediate frames can then be drawn almost mechanically by 'in-betweeners'. Each chief animator could have several in-betweeners working with him[4] to multiply his productivity. (In addition, the tedious task of transferring drawings to cel and colouring them in was also delegated to subordinates.)

In-betweening (which is what in-betweeners do) resembles what mathematicians call *interpolation*: the calculation of values of a function lying in between known points. Interpolation is something that computer programs are very good at, provided the values to be computed and the relationship between them can be expressed numerically. Generally, the relationship between two key frames of a hand-drawn animation is too complex to be reduced to numbers in a way that is amenable to computer processing. But this does not prevent people trying — because of the potential labour savings.

All digital images are represented numerically, in a sense, but the numerical representation of vector images is much simpler than that of bitmapped images, making them more amenable to numerical interpolation. To be more precise, the transformations that can be applied to vector shapes — translation, rotation, scaling, reflection and shearing — are arithmetical operations that can be interpolated. Thus, movement that consists of a combination of these operations can be generated by a process of numerical in-betweening starting from a pair of key frames.

Macromedia's *Flash* is designed for producing simple vector animations, primarily for display on the World Wide Web. Although it can only produce very crude animation, best suited for such ornaments of contemporary civilization as rotating logos and dancing text, it illustrates in a simple form the use of key frames and interpolation as an aid to animation.[5]

3 The mass production approach to animation is almost invariably associated with cartoons featuring characters.

4 This 'him' is not a casual slip: the big cartoon studios of those days did not have what we would consider an enlightened attitude to women as animators.

5 You will also encounter some of the same concepts when we describe time-based multimedia authoring in Chapter 13.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 378
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

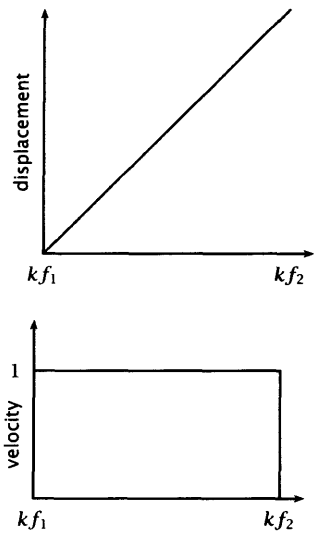


Figure 11.3
Linearly interpolated motion

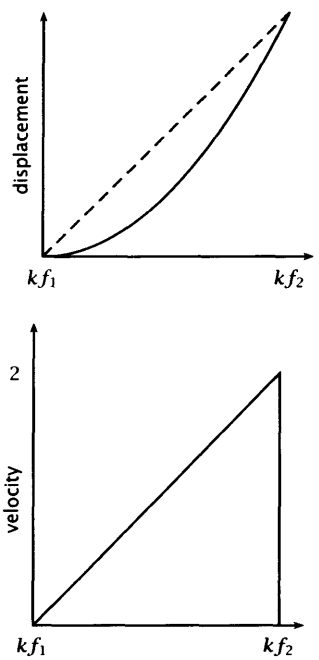


Figure 11.4
Easing in

A Flash animation is organized using a *timeline*, a graphical representation of a sequence of frames, similar to the timeline in video editing applications. Animations *can* be built up a single frame at a time, by inserting individual frames into the timeline, but here we are more interested in inserting just the key frames. A key frame is created simply by clicking in the timeline at the appropriate point and using the `insert keyframe` menu command or its command key equivalent. This just establishes the existence of the key frame; its content must also be added before in-betweening can be specified.

Flash's *stage* is a sub-window in which frames are created by arranging objects, and which is also used to preview animations. Objects can be created on the stage using some built-in drawing tools, similar to but less comprehensive than those of Illustrator or Freehand, or they can be imported from those applications. Bitmap images, in formats including JPEG and PNG, may also be imported and auto-traced to make vector objects; images can be used in bitmap form within a Flash frame, but cannot then be rotated or scaled without potentially degrading the image. Comprehensive support is provided for text; characters in outline fonts can be decomposed into their component outline paths, which can be edited or animated separately.

Graphical objects can be stored in a library in a special form, called a *symbol*, that allows them to be reused. Since interpolated animations, almost by definition, reuse objects, key frames can only be built out of symbols. To animate a symbol, a key frame is selected in the timeline, and the symbol is placed on the stage; the current frame is then moved to the next key frame, and the symbol is moved, scaled, or otherwise transformed, to create the new frame. (The symbol is behaving like a sprite.) Double-clicking anywhere in the timeline between the two key frames brings up a dialogue, which allows you to specify that the in-between frames should be interpolated ('tweened', as Flash puts it). The current frame is moved to the next key frame, if there is one, and the process is repeated. The animation is built up as a sequence of automatically tweened segments, between key frames that have been arranged by hand.

Moving a single symbol about the stage offers little in the way of artistic gratification. Each Flash animation, though, can have an arbitrary number of layers, allowing the independent animation of many symbols. To further ease the work of animating motion, an

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 379
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

object can be moved along a path drawn on a hidden layer; this *motion path* need not be a straight line, so movements that would have to be constructed using many key frames if only rectilinear movements were possible can be achieved in a single tweened sequence with just two key frames. Finally, a symbol can itself be a small animation, that plays within the main movie, so that different motions can be combined. For example, a symbol can consist of an object rotating; this can then be animated on a path, so that the object rotates as it follows the path. The analogy with traditional cel animation — of a basic kind — is fairly complete.

Flash's interpolation is linear. If we just consider interpolated motion in a straight line, this means that a symbol moves an equal distance between each frame, the distance moved being the total distance between its positions in the starting and ending key frames, divided by the number of frames in the sequence. Putting it more simply, the symbol moves at a constant velocity, which causes two problems.

First, motion begins and ends instantaneously, with objects attaining their full velocity as soon as they start to move, and maintaining it until they stop. Nothing really moves like that. To produce a more natural movement, Flash borrows a technique from hand-made animation: the transition from stasis to movement is made more gradual by using smaller, slowly increasing, increments between the first few frames (i.e. the object accelerates from a standstill to its final velocity), a process referred to as *easing in*. The converse process of deceleration is called *easing out*. Figure 11.3 shows the way the horizontal displacement and velocity of an object changes with time when it is moved from an initial position in key frame 1 of $(0,0)$ to a final position in key frame 2 of $(50,50)$, using linear interpolation over 50 frames. Figures 11.4 and 11.5 show how the change might be modified when the motion is eased in or out — we have shown a style of easing that uses quadratic interpolation, that is, the acceleration is constant. More complicated styles are possible and might be preferred. Flash allows the animator to set the degree of easing using a slider that moves from maximum easing in, through a constant velocity, to maximum easing out. In effect, this moves the displacement curve from one like Figure 11.4, via similar curves with less pronounced bulge, through Figure 11.3 and beyond to Figure 11.5. (That is, the acceleration goes from some maximum positive value, through zero, to a maximum negative value.)

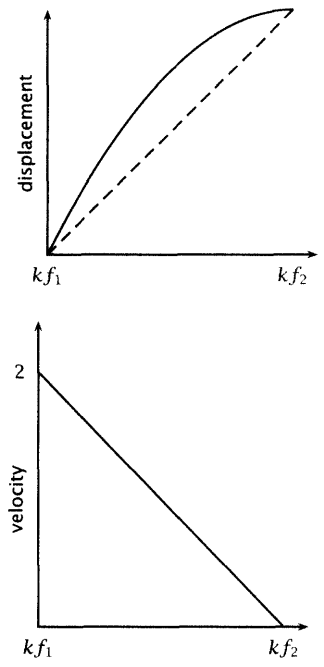


Figure 11.5
Easing out

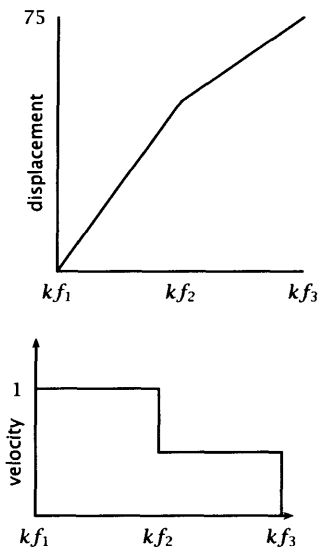


Figure 11.6
Abrupt change of velocity

The second problem with linear interpolation can be seen in Figure 11.6, which shows how displacement and velocity change if we now append to our original sequence a second one of 50 frames, during which our object moves from its position in key frame 2 of (50, 50) to a new position at (75, 75) in key frame 3. Because each sequence is interpolated separately as a straight line, there is a sharp discontinuity at key frame 2; as the velocity graph clearly shows, this will appear as a sudden deceleration at that point in the animation. Again, this is an unnatural sort of movement, that will rarely be what is desired. By clever manipulation of the easing slider, it would be possible to smooth out this abruptness, but a more general solution to the problem is available. In Chapter 4, we stressed that Bézier curves' most attractive property is that they can be joined together smoothly by aligning their tangent vectors. By using Bézier curves instead of straight lines to interpolate between key frames, smooth motion can be achieved. Note that we do not mean that objects should follow Bézier shaped paths, but that the *rate* at which their properties change should be interpolated using a Bézier curve. Flash does not offer the option of any form of non-linear interpolation but other, more elaborate, key frame animation systems do. Although we have only considered displacement in one direction, you should be able to see that similar considerations apply to any transformations that might be interpolated.

Although we claimed earlier that interpolation of vector graphics is easier than interpolation of bitmapped images, you can probably see that there is little advantage, for any but the most basic animation, in using vectors — rather the reverse, because of the stylistic limitation imposed by vector drawing. We will see in the next section that key frame animation can be applied to bitmaps to produce more visually interesting results. The pay-off from using vectors comes in the compact representation that can be used for the final animation. As with vector still image formats, objects can be represented compactly; additionally, as with sprite tracks, their movements can also be represented compactly. For example, the native file format of Flash is *SWF ('Shockwave Flash')*. An SWF file consists of items, which are divided into two broad classes: definitions and control items. The former are used to store definitions of the symbols used in an animation into a dictionary; the latter are instructions to place, remove, or move a symbol (identified by its name in the dictionary). Placement and movement are specified using transformation matrices, so that the position

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 381
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

and any scaling or rotation are specifed at the same time. An SWF file is thus rather like a program, comprising as it does definitions of some objects and instructions that manipulate them. SWF data is encoded in a binary form and compressed, resulting in very small files. For example, an extremely simple tweened animation consisting of twenty frames, during which a red circle moves along a motion path, growing in size as it does so, occupies 450 bytes as an SWF file; the same animation, stored as a QuickTime movie using the Animation compressor (which should be particularly effective on such material) at a medium quality setting, occupies over 19 kbytes.

Motion Graphics

Interpolation between key frames can be applied to bitmapped images. Since bitmaps do not contain identifiable objects, the use of layers to isolate different elements of an animation is essential. The analogy with cel animations is more or less complete — each layer is like a transparent sheet of acetate with something painted on it. Layers can be moved independently, so an animation can be constructed by placing different elements on different layers, and moving or altering the layers between frames. Where the movement or alteration is easily described algorithmically, it can be interpolated between key frames, just as in-betweeners interpolate between a chief animator's key frames. Typically, between key frames, a layer may be moved to a different position, rotated or scaled. These geometrical transformations are easily interpolated, but since we are now concerned with bitmapped images, they may require resampling, and consequently cause a loss of image quality, as we explained in Chapter 5.

After Effects is the leading desktop application for animation of this kind. Because of their shared provenance, After Effects works well in conjunction with Photoshop and Illustrator. A Photoshop image can be imported into After Effects, with all its layers — including adjustment layers — and alpha channels intact; an Illustrator drawing can be imported and rasterized, again with its layers intact. A common mode of working, therefore, is to use the tools and facilities of Photoshop or Illustrator to prepare the elements of an animation on separate layers, and import the result into After Effects where the layers are animated. Photoshop images should be prepared at an appropriate resolution and size for your intended

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 382
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

delivery medium. If they are to be scaled down, they must be large enough to accommodate the maximum reduction that will be applied. Illustrator files can either be rasterized when they are imported and then treated as bitmaps, or they can be continuously rasterized for each frame in the animation. This means that if they are scaled, for example, no detail will be lost.

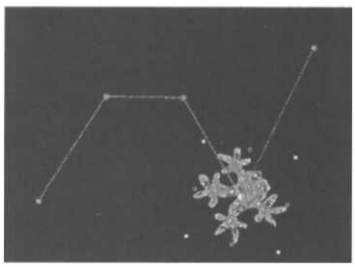


Figure 11.7
Linear spatial interpolation

The simplest animations are made by repositioning layers, either by dragging them or by entering coordinates, and interpolating motion between key frames. After Effects supports both linear and Bézier interpolation, in both space and time. Figure 11.7 shows a tree frog moving along a motion path, defined by linear interpolation between five key frames — the changes in direction are abrupt, as if the frog was a ball bouncing off a hard surface. Figure 11.8 shows the motion path through the same key frames, with Bézier interpolation[6] — the changes in direction are smooth, like a well steered bicycle. These are different forms of spatial interpolation, which are set by moving the layer in the window that shows the image. Temporal interpolation affects the rate of change of position with respect to time. Again, this may be linear, with a constant velocity and instantaneous starting and stopping, as discussed earlier in connection with Flash, or Bézier, where the acceleration is smooth. Temporal interpolation is set in a *value graph* for the corresponding property. The value graphs displayed by After Effects for our frog's velocity with linear and Bézier interpolation are shown in Figures 11.9 and 11.10, respectively. The temporal and spatial interpolation methods are independent: you can use linear temporal interpolation with Bézier motion paths, and vice versa.

6
After Effects actually provides 'auto Bézier' interpolation, where the handles are set automatically on the basis of the neighbouring key frames; 'continuous Bézier' interpolation, where handles are adjusted manually, as in Illustrator, with the direction lines at joins constrained to be the same length and colinear; and 'Bézier' interpolation, where the two direction lines at a join are moved independently.

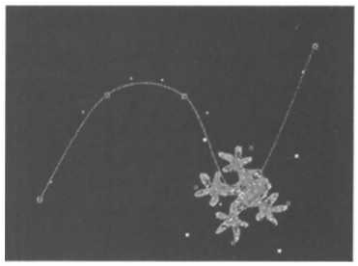


Figure 11.8
Bézier spatial interpolation

Interpolation can be applied to other properties of a layer. In particular, its angle can be varied, so that it appears to rotate. Angles may be set by hand in key frames and interpolated, or the rotation may be determined automatically in conjunction with movement, to maintain the orientation of a layer with respect to its motion path. Scaling, which may be used as a perspective effect to convey the impression of approaching or receding movement, or as a zoom in or out, can also be set in key frames and interpolated.

The degree of control over the interpolation of these spatial properties offered by After Effects is considerable. Using a conventional Bézier pen tool, the graphs showing how a value varies with time may be redrawn. Key frames are inserted automatically when control points are added to the graph. Absolute values may be entered numerically, allowing complete control over positioning

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 383
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

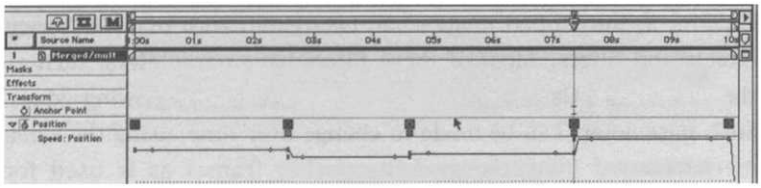


Figure 11.9
Velocity graph for linear temporal interpolation

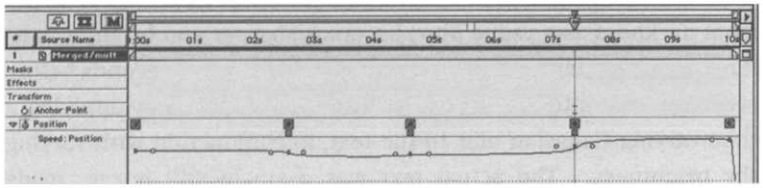


Figure 11.10
Velocity graph for Bézier temporal interpolation

and the rate of movement. Nevertheless, the type of motion that can be produced by interpolating the position, angle, and size of a single layer is restricted. Objects appear to move as a whole, with an unrealistic gliding motion, resembling that seen in simple, low-budget, cut-out animation, as favoured for some pre-school entertainment and education on television. Production of realistic movement — for example, making our frog walk or jump — requires different parts of an object, such as the legs, body, and eyes of a frog, to be placed on separately animated layers. An understanding of the way things move in real life is needed to coordinate the movement of the layers, and there may be little scope for automation.

Key frame animation of bitmapped images is therefore more frequently used for stylized motion. As we mentioned in Chapter 10, travelling mattes are often made by animating a still image in After Effects. Another popular application is the animation of text. Individual characters or words can be placed on layers and animated, just like any other layer, or text may be placed on a path, as in Illustrator, and then moved along that path over time.

The bitmapped representation allows other properties of the image besides its position, angle and size to be altered over time. So, in addition to geometrical transformations, more radical time-based alterations of the layers can be achieved. As we described in

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 384
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043



Figure 11.11
Frames from a title sequence

Chapter 5, bitmapped images can be treated with many different effects and filters. Most of these filters have parameters, such as the radius of a Gaussian blur, or the brightness of glowing edges. Such parameters can be made to change over time, using the same mechanism of interpolation between key frames as is used for interpolating motion. Doing so allows some unique effects to be generated.

For example, Figure 11.11 shows a sequence of frames extracted from the title sequence of a short film. The title, 'part of', emerges from darkness as a vague blur, becomes sharper and brighter until it reaches maximum clarity, where it is held for a few seconds before receding back into the darkness. This was achieved by applying a time-varying Gaussian blur to the text, in conjunction with varying the brightness. The actual text was a single still image, made in Photoshop, that was otherwise unaltered. Figure 11.12 shows the graphs of the Gaussian blur and brightness values that were used. The blur starts with a very high value, which renders the text illegible; in the same key frame, the brightness is substantially reduced. The values are interpolated to a point where the blur is removed and the brightness brought right up. Bézier interpolation is used to ensure a smooth fade up. The values are held constant between the middle two key frames, as shown by the flat portions of the graphs, and then, to make the title fade back into nothing, a symmetrical interpolation is used to a final key frame where the values are identical to those at the beginning.

The effects that can be achieved using time-varying filters on bitmapped images have more in common with graphic design than with mainstream cartoons or art animations. They are often known by the more suggestive name of *motion graphics*. Many of the techniques first appeared in title sequences for feature films, and credit sequences remain a typical application.

Motion graphic effects can be applied to live video as well as to still images. That is, a video clip can be moved across the screen, scaled and rotated, and treated with time-varying filters. By arranging clips on separately animated layers, elaborate compositions can be constructed. The result is fast becoming the time-based equivalent of the Layered Look in graphic design.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 385
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

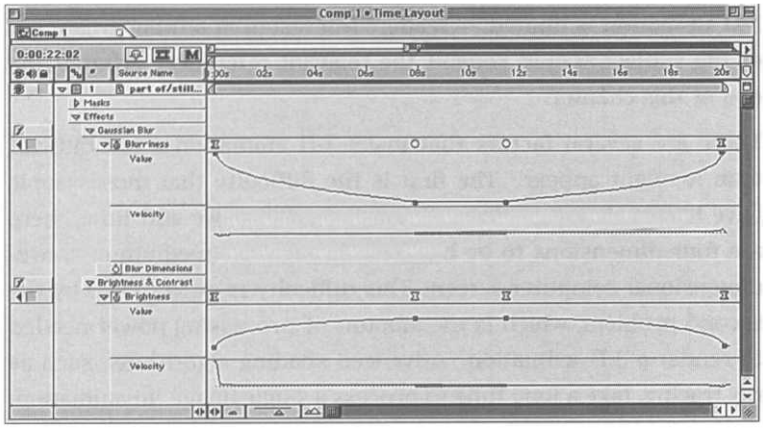


Figure 11.12
Value graphs used for the title sequence

3-D Animation

3-D animation is easy to describe, but much harder to do. No new concepts beyond those already introduced in this chapter and in Chapter 4 are needed to understand the essence of the process. The properties of 3-D models are defined by numerical quantities. Changing the numbers changes properties such as an object's position in space, its rotation, its surface characteristics, and even its shape. The intensity and direction of light sources and the position and orientation of a camera are also numerically defined. In order to animate a three-dimensional scene, therefore, all that is necessary is to set up an initial scene and render it as the first frame of the animation, make some changes to parameters, render the next frame, and so on. Because the values that are being changed are numerical, some kinds of change can be interpolated between key frames; a time line can be used as a convenient way of organizing the animation, and motion paths in three dimensions (often 3-D Bézier splines) can be used to describe movement.

Whereas simple 3-D animations, such as tumbling logos and rotating globes, really can be made very easily — and there are dedicated packages available for such tasks — high-quality photo-realistic animations, such as those employed in television advertisements, music videos, and film special effects, require huge resources: time, processor power and memory, dedicated software, and above all, highly skilled specialized personnel. Multimedia production can rarely afford these resources. For this reason our description of

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 386
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

3-D animation is limited — readers interested in a fuller treatment of the subject should consult the relevant references given at the end of this chapter.

There are several factors that make 3-D animation more difficult than it might appear. The first is the difficulty that most people have in visualizing in three dimensions. When we add time, there are four dimensions to be handled through the medium of a two-dimensional computer screen. This difficulty is exacerbated by the second problem, which is the amount of processing power needed to render a 3-D animation. Advanced shading algorithms, such as ray tracing, take a long time to process a single image. In animation, we need at least twelve, and up to thirty, images to be processed for every second of completed animation. This makes generating fully rendered previews impossible on anything but the most powerful workstations or networks of distributed processors.

Large budgets, patience and practice can overcome these problems, but what remains is the necessity to change a large number of parameters in such a way as to produce convincing movements. At the very highest level of 3-D computer-generated animation, the solution that is adopted is to provide a rich interface giving the animator complete control over movement. For animating figures, this resembles the type of control used by a puppeteer to control a mannequin — body suits equipped with motion sensors, like those used to control animatronic puppets, are even used sometimes.[7] This is in marked contrast to the approach taken to animation by computer programmers, whose well-trained instinct is to try to automate everything. In an attempt to overcome the limitations of simple interpolation schemes, considerable research efforts have been expended on ways of producing convincing motion in three dimensions automatically.

7
At this level, computer-generated animation is certainly no simpler than traditional animation, and is only used in order to produce results that cannot be achieved economically using physical means.

One of the key approaches is to provide certain kinds of behaviour that can be applied to objects, and the way they interact. A simple type of behaviour consists of making one object point at another. This is most useful when the pointing object is a camera or light. If a camera is pointed at an object, it will maintain that object in its field of view, no matter where it moves; a spotlight pointed at an object will automatically follow it, as a real spotlight follows a dancer on a stage, for example. Actual objects in the scene can also be made to point at each other: a sunflower can be made to always point at the sun, for example. A variation on this behaviour is to have one object track another, i.e. follow its motion at a distance. This can be used

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 387
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

crudely to animate chase scenes. Like pointing, it can be applied to a camera, allowing it to follow an object or character through a scene, even in places where it would be physically impossible for a real camera to go.

Some 3-D animation systems incorporate behaviour based on the physical laws of motion. For example, they allow the user to specify that an object should accelerate from zero under the influence of an external force whose magnitude is specified as a parameter. Taking this further, moving objects can be made to collide realistically, or bounce off solid surfaces. These behaviours are based on simple laws of physics that encapsulate the possible motion in a few equations. Unfortunately, many realistic types of movement cannot be so easily described, and other methods must be used.

Kinematics is the study of the motion of bodies without reference to mass or force. That is, it is only concerned with how things can move, rather than what makes them do so. In animation, it is most useful in connection with jointed structures, such as the limbs of human or animal figures. Because they are joined together, the various parts of your arm, for example, can only move in certain ways. To produce realistic movement, a 3-D model of an arm must obey the same *kinematic constraints* as a real arm: if the upper arm is raised, the lower arm and hand must come with it, for example. Whereas, in reality, it is the motion of the upper arm that propels the rest of the arm, in an animation system, modelling movement in this way — from the beginning of a chain of connected elements to the end — is not very helpful to the animator. It is more useful to be able to position the object which is at the end of the chain — a hand, say — and then make the other elements — the rest of the arm — move to accommodate it. It is usually the extremities that impose limitations on movement; when walking, a foot must stay above the ground, resting on it at each step, for example. It takes considerable understanding of the way limbs move to ensure that this will happen correctly by moving the thigh, so it is preferable for the software to work out the movements of the leg from the animator's placement of the foot, and so on. This type of modelling is called *inverse kinematics*, since it works backwards from effect to cause. Inverse kinematics can be applied to any structure that is modelled as a linked chain of objects. It is routinely provided by 3-D animation programs that support such structures. Poser, for example, can be set to automatically apply inverse kinematics to the arms and legs of figures.

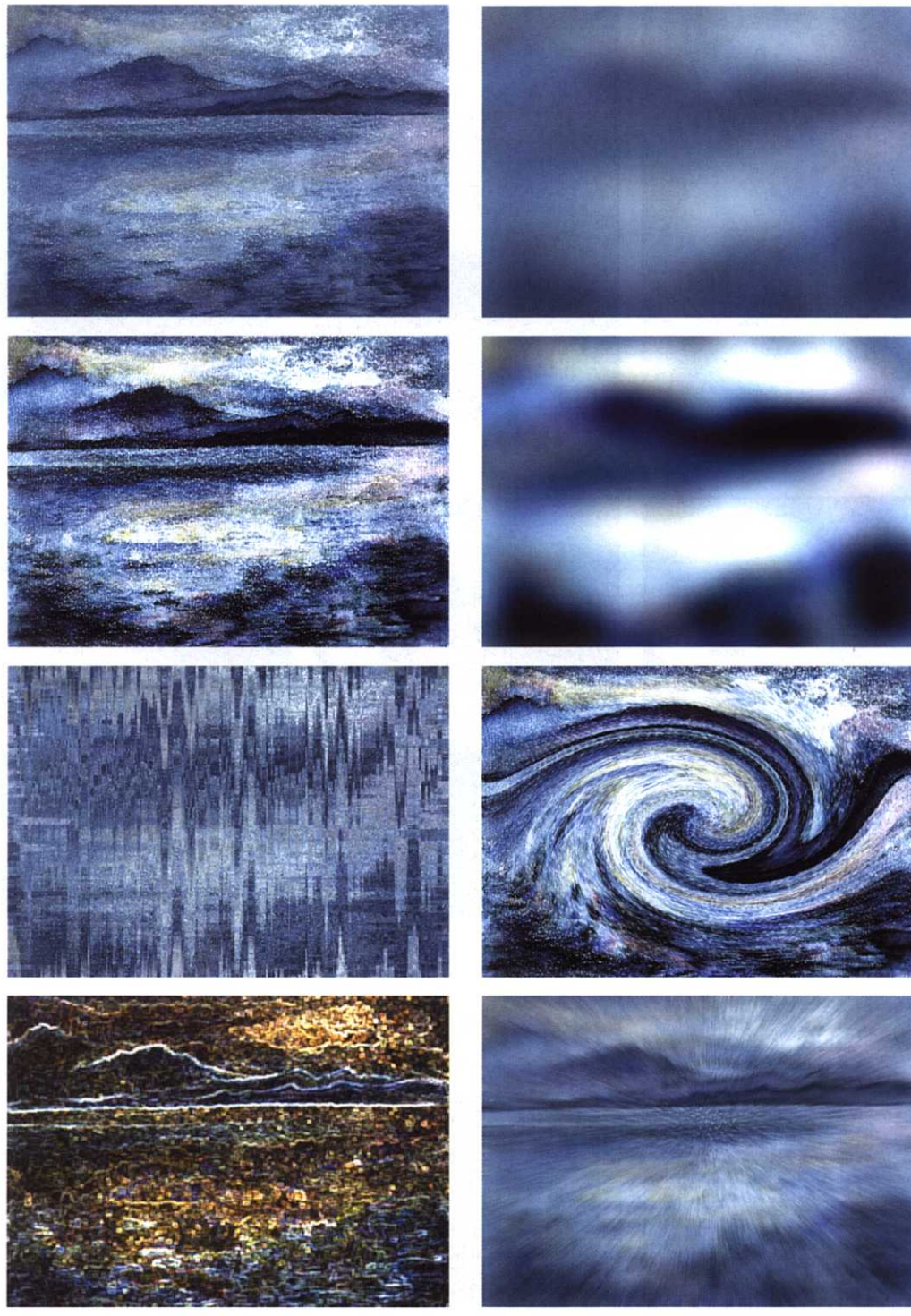
DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 388
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

A little experimentation will show that computation of movement using inverse kinematics is not entirely straightforward. In particular, the kinematic constraints on arms and legs do not uniquely determine the possible movements that the limbs can make to accommodate movements of the extremities. Try lifting your right hand to touch the tip of your nose with your first finger. How many different ways can your right elbow move while you do so? In order to fix a particular type of movement, extra constraints, such as minimizing the potential energy of the whole structure, must be added. To produce movements that defy these constraints, while still being physically possible, inverse kinematics must be abandoned, and the parts must be positioned by hand.

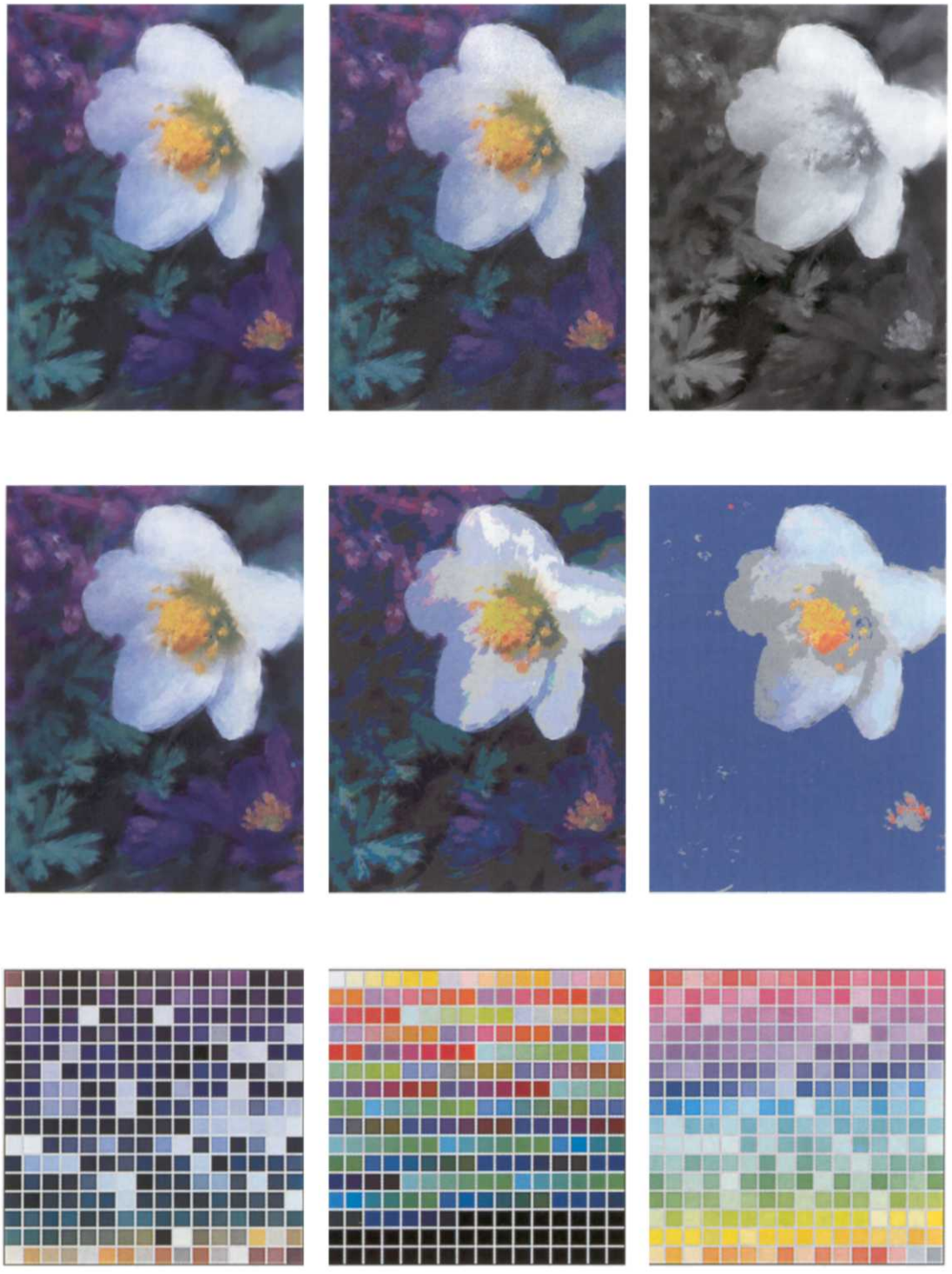
Despite what we said earlier, there are some types of 3-D animation that can be done effectively using simple interpolation of objects' properties. As you might expect from our remarks in Chapter 4, specialized applications, such as Bryce, are generally the most succesful. Bryce does allow basic animation of the motion of objects, by interpolating between key frames — the interface is very similar in spirit to that used in After Effects for animating layers, with Bézier motion paths and velocity graphs that can be modified. The most effective application is animating a camera to obtain a 'fly-through' of a scene; the camera can optionally be made to point at an object while it follows its motion path. This inherently artificial type of motion is much more effective when interpolated than more natural movements can be.

Bryce also includes some specialized animation techniques that are suited to its particular domain of terrain modelling. Most aspects of the sky can be animated, to produce weather effects and to model the passage of time. Clouds can be made to move, as if by the wind, and their shape, density and colour can be changed, to indicate a gathering storm, for example. Rainbows can be made to appear and fade; fog can swirl and dissolve. The sun and moon can rise and set; as the sun passes overhead, shadows will automatically move correctly. Even geological processes can be modelled: mountains can be made to erode into hills, and a landscape can be shaken by an earthquake. All of these effects can be achieved with relative simplicity by animating properties of the scene using key frames and interpolation.

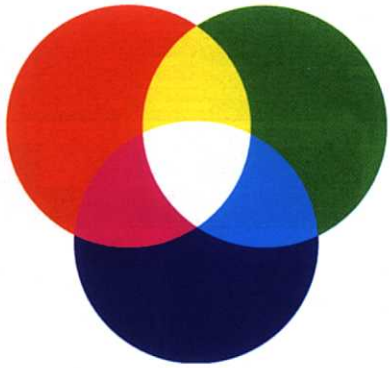
DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 389
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043



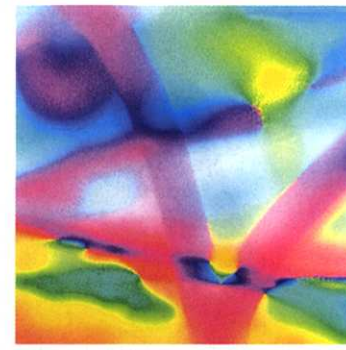
14. Filters applied to a bitmapped image



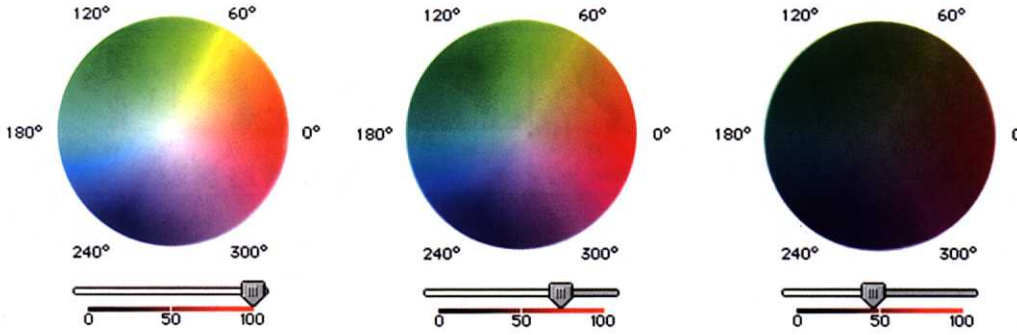
15. Indexed colour



16. Complementary colours



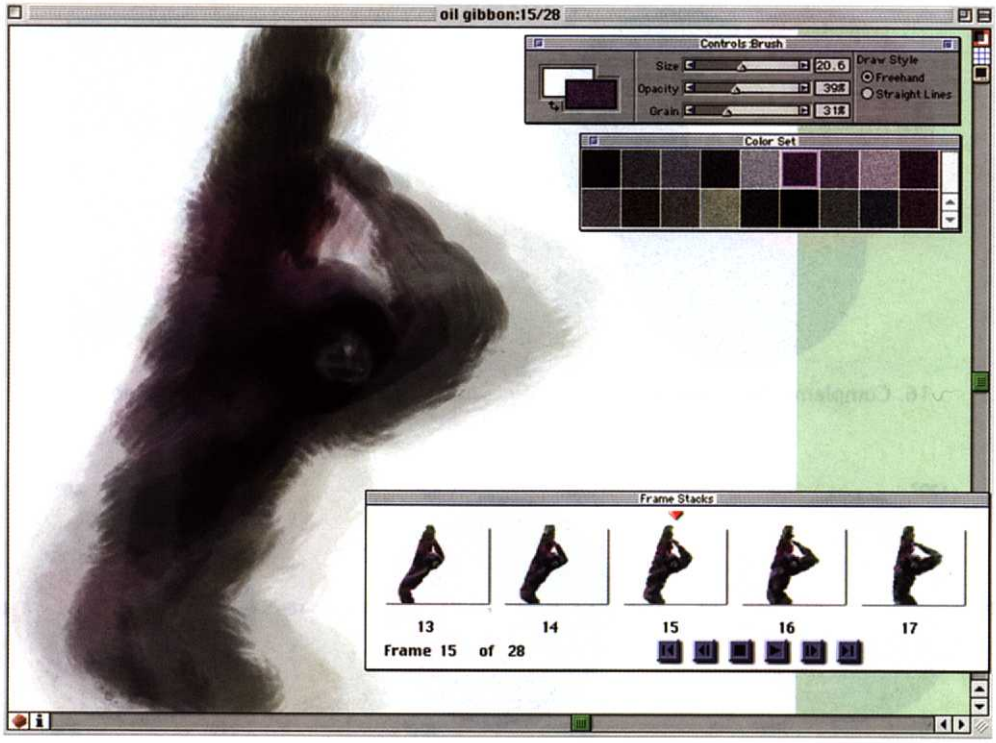
17. Subtractive colour mixing



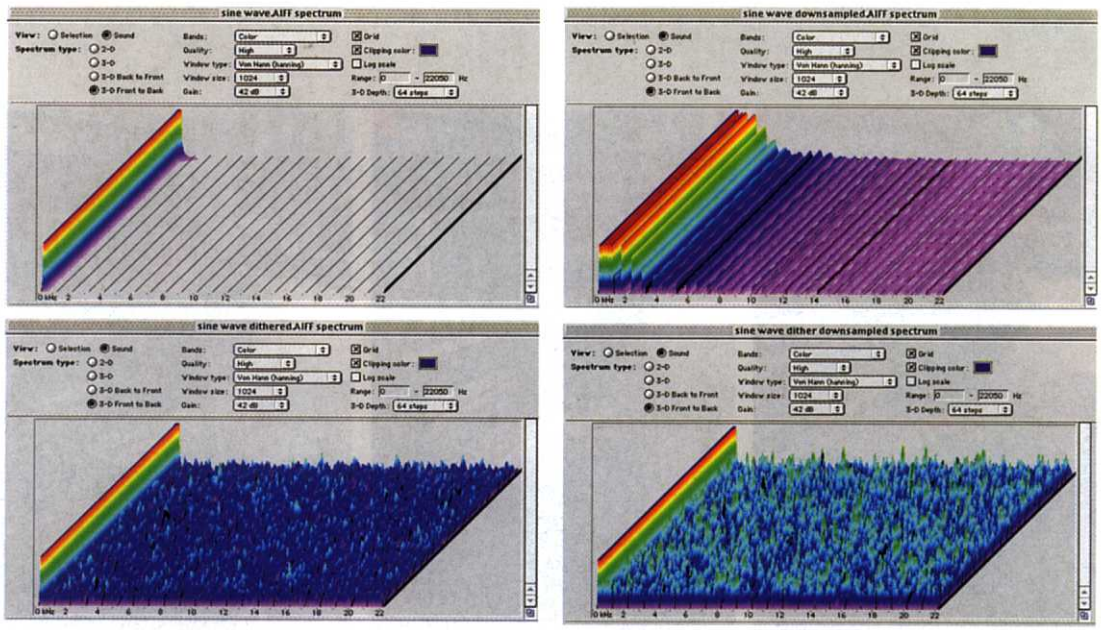
18. Hue and saturation



19. Colour correction



20. Animating using a framestack



21. Effect of sampling and dithering on audio frequency spectrum

Virtual Reality

Originally, the phrase 'virtual reality' was used to describe an immersive sensory experience of a synthetic world. Head-mounted displays, which are sensitive to head movements, are used to project images on the user's eyes, modifying them as the head is moved, so that the user appears to be inside a 3-D world, looking around. Data gloves track hand movements, allowing the display to incorporate an image of the user's arm; haptic interfaces provide tactile feedback, so that users can touch and feel objects in the virtual world. Taken to the extreme, virtual reality of this sort would be the ultimate in multimedia, stimulating all the senses at once.

The high cost of the interface hardware required by immersive virtual reality (together with the understandable reluctance on the part of most of the adult population to immerse their body in strange electronic devices) has confined it to flight and industrial simulations, and specialist games arcades. A more modest vision of virtual reality (VR), as 3-D graphics that can be explored, has evolved. Even this version of VR has not yet achieved widespread acceptance, largely because the heavy demands it makes on processor power lead to disappointing results on desktop computers. Two VR technologies deserve a brief mention, since they can be incorporated in Web pages and other multimedia productions with some success, and promise to become more important as computer power catches up with the vision of VR enthusiasts.

VRML

The *Virtual Reality Modeling Language (VRML)* was created on a wave of enthusiasm for VR and the World Wide Web in 1994. The intention was to provide a mechanism for distributing virtual worlds over the Internet, using Web browsers as the interface. To this end, VRML was a text-based language, that allowed 3-D objects and scenes to be described in a programming language-like notation. VRML 1.0 did little more; the main additional feature was the capability of embedding hyperlinks in scenes, using URLs. Subsequently VRML 2.0 added support for interactivity, via scripting of the sort to be described in Chapter 14, and allowed for video and audio to be embedded in VRML worlds (so, for example, a television

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 394
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

set could be made to show a movie). VRML became an ISO standard[8] in December 1997.

VRML allows the specification of objects, in terms of their geometry (whether they are a cube, cylinder, sphere, and so on) and the material of which they are composed. Textures can be mapped onto the surfaces of objects, which can be placed in 3-D space using transformations. Scenes can be lit in a variety of ways, by specifying the type and position of light objects. The basic language thus provides a way of describing the sorts of scenes that can be constructed with conventional 3-D modelling programs, although it lacks some of the more elaborate features, such as NURBS and metaballs. The description is explicit: for example, a terrain might be modelled as an elevation grid, a type of VRML object that specifies a set of points, forming a grid, each at a different height. In VRML, the dimensions of the grid and the height of each grid point must be explicitly specified. Constructing VRML scenes by hand is thus a painstaking and error-prone business. Most 3-D modellers will generate VRML as output from the normal interactive modelling tools, however, which provides an easier way of constructing scenes.

It might appear that VRML is no more than an alternative representation of the output of a modeller, and, as far as the language goes, this is more or less the case. It does, as we noted earlier, provide additional features for interactivity and embedding multimedia, but the main distinctive feature lies not so much in the language as in the way it is displayed. Once a VRML file has been downloaded into a suitable browser — either a Web browser with an appropriate plug-in, or a dedicated VRML browser — the user can explore the 3-D world it describes. That is, they can move the viewpoint through the space of the model, as if they were moving about in it.[9] To that extent, VRML deserves to be considered a form of virtual reality.

To create the illusion of moving through a 3-D space, VRML must be rendered in real-time. As we have stated several times, realistic rendering of 3-D models is a computationally intensive task, which is usually only feasible with special hardware, such as 3-D accelerator cards. This is one of the main obstacles to the widespread use of VRML, although a lack of commitment to the format by major software vendors may be more significant. At the time of writing, a host of competing — mostly proprietary — formats for delivering 3-D models over the World Wide Web is available, with none, as yet, achieving any widespread use.

8 ISO/IEC-14772-1:1997. The language defined by the standard is identical to VRML 2.0, but is sometimes referred to as VRML 97.

9 The experience will, of course, be familiar to players of certain types of computer game.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 395
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

QuickTime VR

QuickTime VR (or QTVR, for short), part of QuickTime, offers a very basic VR experience. There are two types of QuickTime VR movies: panoramic movies and object movies. The former presents a 360° view of a scene — the interior of a room, or a valley surrounded by mountains, for example. Using their mouse, a user can drag the scene, as if looking around. It is also possible to zoom in or out, in order to view the scene in more or less detail. Object movies, in contrast, allow the user to examine an object from different angles, as if by walking round it, again by dragging with the mouse. QTVR movies of either sort may contain *hot spots*, which are active areas that contain links to other movies. Clicking on a hot spot causes the linked movie to replace the current one. A typical use of hot spots is to allow a user to go through a door from one room into another.

QTVR movies can be generated from some 3-D programs, such as Bryce. They can also be made from photographs, allowing them to represent real scenes and objects. To achieve the best results, a special rotating rig is used to hold the camera (for panoramas) or an object. A sequence of pictures is taken, with the rig being rotated a fixed amount between each picture. These are then scanned (or read in to the computer if a digital camera is being used) and special software 'stitches' them together and renders the result as QTVR. The purpose of the rig is to ensure that the individual pictures fit together perfectly. If an ordinary tripod is used for panoramas, or a record turntable or similar device for object movies, there may be discontinuities; stitching software will attempt to compensate for these, but the result may be distortion of the images.

Since QTVR is part of QuickTime, panoramas and object movies can be combined with audio and video. Most usefully, they can be viewed by any software that uses QuickTime; in particular, the QuickTime plug-in for Web browsers allows QTVR to be embedded in Web pages, in the same way as video clips can be (see Chapter 13).

QuickTime VR and VRML might be considered travesties of the original vision of immersive virtual reality, but they have the advantage of being implementable without special interface devices or powerful workstations. They offer the possibility of new approaches to interfaces to multimedia, based on the organization of media in three-dimensional spaces.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 396
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Further Information

[Lay98] is the best starting point for learning more about all aspects of animation. [Vin92] and [O'R98] deal specifically with computer-generated 3-D animation, and [WW92] describes the algorithms it employs. [HW96] is a full description of VRML. [McC96] offers some interesting thoughts about VR and user interfaces.

Exercises

1. What are the advantages and disadvantages of using a scanner or a digital stills camera to capture traditional art work as animation sequences? For what types of animation would you have to use a video camera connected to a computer?

2. How could you incorporate drawn animation into a live-action video sequence without using a special effects program?

3. If an animation sequence is to be saved as QuickTime, what factors will influence your choice of codec? Under what circumstances would it be appropriate to treat the animated sequence exactly like a live-action video sequence?

4. When would it be appropriate to use an animated GIF for an animation sequence? What problems are associated with animated GIFs?

5. In what ways is a sprite animation track radically different from a video track containing animation?

6. For what type of work would sprite animation be particularly suitable and why?

7. In the example of a planet moving across a starscape, how could you add rotation of the planet on its axis to the sprite animation?

8. What problems are associated with using basic linear interpolation to do 'in-betweening' between key frames?

9. How would you use Flash's easing facility to set up a movement that eases in *and* out?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 397
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

10. Describe the motion of an object whose position is animated in After Effects using Bézier interpolation for the motion path, and linear interpolation for the velocity.

11. Create a very simple title for a video clip as a single image in a bitmapped graphics application such as Photoshop or Painter, and save it as a still image file. Using whatever tools are available (Premiere, After Effects, etc.), create a pleasing ten second title sequence by simply applying time-varying effects and filters to this single image. (If you want to go for a more sophisticated result, and have the necessary tools, you might create your original image on several layers and animate them separately.)

12. Do the models generated by 3-D applications contain enough information to be used in conjunction with a haptic interface to provide tactile feedback to a user? If not, what extra information is needed?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 398
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

12 Sound

Sound is different in kind from any of the other digital media types we have considered. All other media are primarily visual,[1] being perceived through our eyes, while sound is perceived through the different sense of hearing. Our ears detect vibrations in the air in a completely different way from that in which our eyes detect light, and our brains respond differently to the resulting nerve impulses. Sound does have much in common with one other topic we have considered, though. Although sound is, for most of us, a familiar everyday phenomenon, like colour, it is a complex mixture of physical and psychological factors, which is difficult to model accurately.

Another feature that sound has in common with colour is that you may not always need it. Whereas a multimedia encyclopedia of musical instruments will be vastly enriched by the addition of recordings of each instrument, few, if any, Web pages need to play a fanfare every time they are visited. Sounds can be peculiarly irritating; even one's favourite pieces of music can become a jarring and unwelcome intrusion on the ears when inflicted repeatedly by a neighbour's sound system. Almost everyone has at some time been infuriated by the electronic noises of a portable games console, the cuter varieties of ringing tone of a mobile phone, or the rhythmic hiss that leaks out of the headphones of a personal stereo. The thoughtless use of such devices has become a fact of modern life; a similar thoughtlessness in the use of sound in multimedia

1
Text may exceptionally be rendered in other ways, but the graphic representation is the norm.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 399
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

productions should be avoided. At the very least, it should always be possible for users to turn the sound off.

Some users, though, don't need to be able to turn sounds off, because they can't hear them anyway. Not only are some people unable to hear, many others use computers that are not equipped to reproduce sound. Although the Multimedia PC specification[2] requires a sound card (and all Macintosh computers have sound capabilities), older PCs, and those used in the business environment, are rarely fitted with them. It is always considerate to provide some alternative to sound, such as captions or transcripts of speech, for the benefit of those who cannot hear. If you know that your multimedia production is destined to be used in an environment where sound hardware is not typically available, then it may be advisable to avoid the use of sound altogether.

2
See page 45.

There are two types of sound that are special: music and speech. These are also the most commonly used types of sound in multimedia productions. The cultural status of music, and the linguistic content of speech mean that these two varieties of sound function in a different way from other sounds and noises, and play special rôles in multimedia. Representations specific to music and speech have been developed, to take advantage of their unique characteristics. In particular, compression algorithms tailored to speech are often employed, while music is sometimes represented not as sound, but as instructions for playing virtual instruments.

The Nature of Sound

If a tuning fork is struck sharply on a hard surface, the tines will vibrate at a precise frequency. As they move backwards and forwards, the air is compressed and rarefied in time with the vibrations. Interactions between adjacent air molecules cause this periodic pressure fluctuation to be propagated as a wave. When the sound wave reaches the ear, it causes the eardrum to vibrate at the same frequency. The vibration is then transmitted through the mechanism of the inner ear, and converted into nerve impulses, which we interpret as the sound of the pure tone produced by the tuning fork.

All sounds are produced by the conversion of energy into vibrations in the air or some other elastic medium. Generally, the entire

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 400
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

process may involve several steps, in which the energy may be converted into different forms. For example, if one of the strings of an acoustic guitar is picked with a plectrum, the kinetic energy of the musician's hand is converted to a vibration in the string, which is then transmitted via the bridge of the instrument to the resonant cavity of its body, where it is amplified and enriched by the distinctive resonances of the guitar, and then transmitted through the sound hole. If one of the strings of an electric guitar is picked instead, the vibration of the string as it passes through the magnetic fields of the pickups induces fluctuations in the current which is sent through the guitar lead to an amplifier, where it is amplified and used to drive a loudspeaker. Variations in the signal sent to the speaker coil cause magnetic variations, which are used to drive the speaker cone, which then behaves as a sound source, compressing and rarefying the adjacent air.

While the tines of a good tuning fork vibrate cleanly at a single frequency, most other sound sources vibrate in more complicated ways, giving rise to the rich variety of sounds and noises we are familiar with. As we mentioned in Chapter 2, a single note, such as that produced by a guitar string, is composed of several components, at frequencies that are multiples of the fundamental pitch of the note. Some percussive sounds and most natural sounds do not even have a single identifiable fundamental frequency, but can still be decomposed into a collection — often a very complex one — of frequency components. As in the general case of representing a signal in the frequency domain, which we described in Chapter 2, we refer to a sound's description in terms of the relative amplitudes of its frequency components as its frequency spectrum.

The human ear is generally considered to be able to detect frequencies in the range between 20 Hz and 20 kHz, although individuals' frequency responses vary greatly. In particular, the upper limit decreases fairly rapidly with increasing age: few adults can hear sounds as high as 20 kHz, although children can. Frequencies at the top end of the range generally only occur as components of the transient attack of sounds. (The general rule that high frequencies are associated with abrupt transitions applies here.) The highest note on an ordinary piano — which more or less defines the limit of most Western music — has a fundamental frequency of only 4186 Hz when in concert pitch.[3] However, it is the transient behaviour of notes that contributes most to the distinctive timbre

3 That is, using even temperament, with the A above middle C equal to 440 Hz.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 401
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

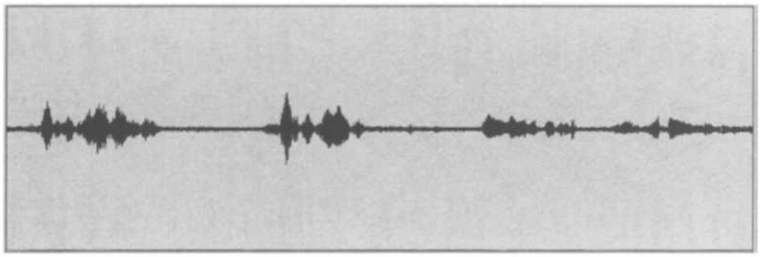


Figure 12.1
'Feisty teenager'

of instruments: if the attack portion is removed from recordings of an oboe, violin, and soprano playing or singing the same note, the steady portions are indistinguishable.

Interesting sounds change over time. As we just observed, a single musical note has a distinctive attack, and subsequently it will decay, changing its frequency spectrum first as it grows, and then as it dies away. Sounds that extend over longer periods of time, such as speech or music, exhibit a constantly changing frequency spectrum. We can display the *waveform* of any sound by plotting its amplitude against time. Examination of waveforms can help us characterize certain types of sound.

➲ The idea of a sound's frequency spectrum changing might be slightly confusing, if you accept that any complex waveform is built out of a collection of frequency components. Strictly, Fourier analysis (as introduced in Chapter 2) can only be applied to *periodic* signals (i.e. ones that repeat indefinitely). When analysing signals with a finite duration, various expedients must be adopted to fit into the analytic framework. One approach is to treat the entirety of a signal as one cycle of a periodic waveform; this is roughly what is done when images are broken down into their frequency components. An alternative is to use a brief section of the signal as if it were a cycle, thus obtaining a snapshot of the frequency make-up at one point. For audio signals, this provides more useful information. A spectrum analysis is typically obtained by sliding a window through the waveform to obtain a sequence of spectra, showing how the signal's frequency components change over time.

Figures 12.1 to 12.7 show waveforms for a range of types of sound. Figure 12.1 is a short example of speech: the main speaker repeats the phrase 'Feisty teenager' twice, then a more distant voice responds. You can clearly identify the syllables, and recognize that the same phrase is repeated, the second time faster and with more emphasis. In between the phrases there is almost silence —

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 402
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

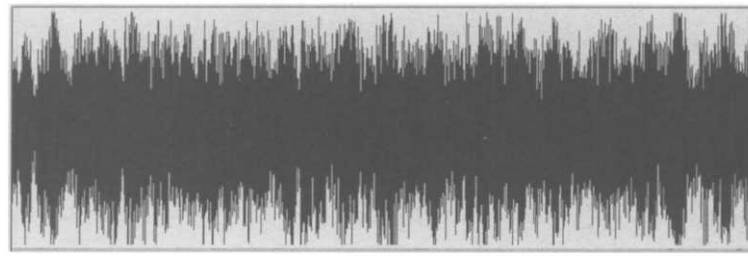


Figure 12.2
Didgeridoo



Figure 12.3
Boogie-woogie

the sound was recorded in the open air and there is background noise, which is visible as the thin band running along the axis. You can see that it could be possible to extract individual syllables and recombine them to synthesize new words, and that, if it were necessary to compress speech, a lot could be achieved by removing the silences between phrases. The clearly demarcated syllables also provide a good basis for synchronizing sound with video, as we will see later.

The next four figures show the waveforms of some different types of music. The first three are purely instrumental, and do not exhibit the same character as speech. The first is taken from an Australian aboriginal didgeridoo piece. This is characterized by a continuous drone, which requires the musician to employ a 'circular breathing' technique to maintain it. The waveform shows this drone, as the thick continuous black region, with its rhythmic modulation. Figure 12.3 shows the waveform of a piece of boogie-woogie, played by a pianist accompanied by a small group. The rhythm is clearly visible, but it is not possible to distinguish the melody played in the right hand (unless, perhaps, you are a very experienced audio technician). Figure 12.4 is a completely different waveform, corresponding to a very different piece of music: a contemporary work arranged for violin, cello, and piano. It shows

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 403
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

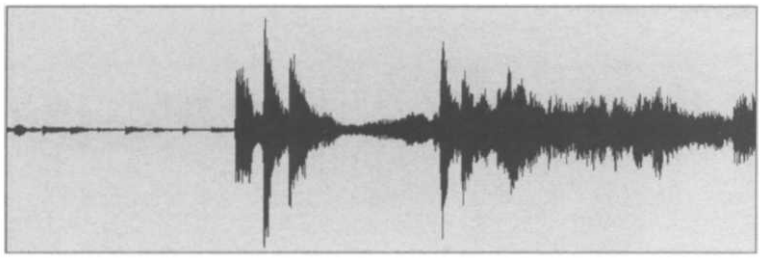


Figure 12.4
Violin, cello and piano



Figure 12.5
'Men grow cold...'

a great dynamic range (difference between the loudest and quietest sounds). Although the steep attack of the louder phrases tells you something about the likely sound of this music, there is no obvious rhythm, and it is not possible to pick out the different instruments (although they can be clearly identified when listening to the music).

As you would expect, singing combines characteristics of speech and music. Figure 12.5 is typical: the syllables of each word are easily identifiable, as is the rhythm, but the gaps between sung phrases are filled with the musical accompaniment. It is possible to see the singer's phrasing, but quite impossible to deduce the lyrics,[4] and, although voice prints are unique to each individual, we doubt whether any readers could identify the singer from this waveform, despite her distinctive voice. (It's Marilyn Monroe.)

Figures 12.6 and 12.7 are both natural water sounds. The first is a recording of the trickling sound of water in a small stream; it is almost continuous. The random spikes do not correspond to any audible clicks or other abrupt sound, they are just slight variations in the water's flow, and some background noise. The second waveform was recorded on the seashore. There is a constant background of surf and two distinct events. The first is a wave breaking fairly close to the microphone, while the second is the

4 *Men grow cold, as girls grow old/And we all lose our charms in the end.*

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 404
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

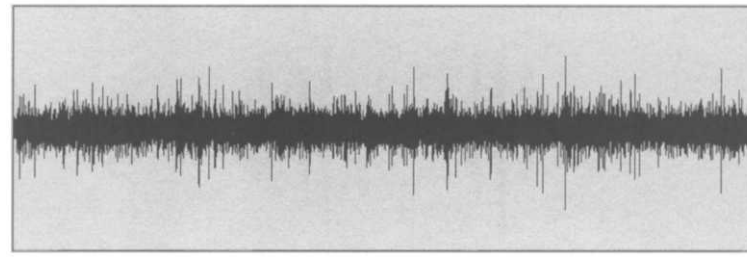


Figure 12.6
A trickling stream

water splashing into a nearby rock pool and then receding through a gap in the rocks. This waveform can almost be read as a story.

As these illustrations show, a waveform display can show a certain amount of the gross character of a sound, but it does not convey the details, and it is not always easy to correlate against the sound as it is heard. The main advantage of these visual displays is their static nature. A piece of sound can be seen in its entirety at one time, with relationships such as the intervals between syllables or musical beats visible. This makes analysis of the sound's temporal structure — which is especially useful for synchronization purposes — relatively simple, compared to performing the same analysis on the dynamically changing sound itself, which is only heard an instant at a time.

Waveforms and the physics of sound are only part of the story. Sound only truly exists as a sensation in the mind, and the perception of sound is not a simple registering of the physical characteristics of the waves reaching the ears. Proofs of this abound, both in the literature and in everyday experience. For example, if a pure 200 Hz tone is played, first softly, then louder, most listeners will believe that the louder tone has a lower pitch than the quieter one, although the same illusion is not perceived with higher frequency tones. Similarly, complex tones sometimes seem to have a lower pitch than pure tones of the same frequency. Most people with good hearing can distinguish the sound of their own name spoken on the opposite side of a noisy room, even if the rest of what is said is inaudible, or carry on a successful conversation with someone speaking at a volume lower than that of the ambient noise.

One of the most useful illusions in sound perception is stereophony. The brain identifies the source of a sound on the basis of the differences in intensity and phase between the signals received from

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 405
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

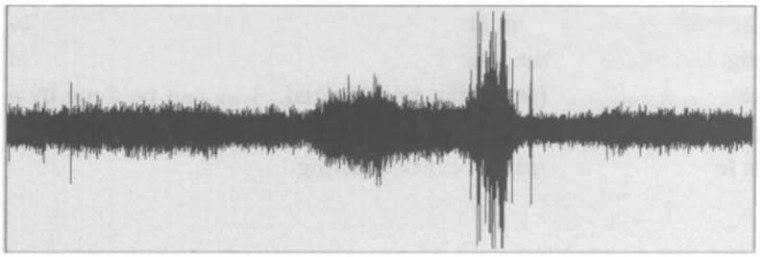


Figure 12.7
The sea

the left and right ears. If identical signals are sent to both ears, the brain interprets the sound as coming from a non-existent source that lies straight ahead. By extension, if a sound is recorded using a pair of microphones to produce two monophonic channels, which are then fed to two speakers that are a suitable distance apart, the apparent location of the sound will depend on the relative intensity of the two channels: if they are equal it will appear in the middle, if the left channel is louder (because the original sound source was nearer to the left hand microphone) it will appear to the left, and so on. In this way, the familiar illusion of a sound stage between the speakers is constructed.

Because of the psychological dimension of sound, it is unwise, when considering its digitization and reproduction, to place too much reliance on mathematics and measurable quantities. Pohlmann's comments[5] about the nature of sound and its reproduction should be borne in mind:

> "Given the evident complexity of acoustical signals, it would be naive to believe that analog or digital technologies are sufficiently advanced to capture fully and convey the complete listening experience. To complicate matters, the precise limits of human perception are not known. One thing is certain: at best, even with the most sophisticated technology, what we hear being reproduced through an audio system is an approximation of the actual sound."

5 [Poh95], p 5.

Digitizing Sound

The digitization of sound is a fairly straightforward example of the processes of quantization and sampling described in Chapter 2.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 406
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Since these operations are carried out in electronic analogue to digital converters, the sound information must be converted to an electrical signal before it can be digitized. This can be done by a microphone or other transducer, such as a guitar pickup, just as it is for analogue recording or broadcasting.

Sampling

A sampling rate must be chosen that will preserve at least the full range of audible frequencies, if high-fidelity reproduction is desired. If the limit of hearing is taken to be 20 kHz[6], a minimum rate of 40 kHz is required by the Sampling Theorem. The sampling rate used for audio CDs is 44.1 kHz — the precise figure being chosen by manufacturers to produce a desired playing time[7] given the size of the medium. Because of the ubiquity of the audio CD, the same rate is commonly used by the sound cards fitted to computers, to provide compatibility. Where a lower sound quality is acceptable, or is demanded by limited bandwidth, sub-multiples of 44.1 kHz are used: 22.05 kHz is commonly used for audio destined for delivery over the Internet, while 11.025 kHz is sometimes used for speech. Another important sampling rate is that used by DAT (digital audio tape) recorders, and also supported by the better sound cards. Although these commonly offer a variety of sampling rates, 48 kHz is used when the best quality is desired. DAT is a very suitable medium for live recording and low budget studio work, and is often used for capturing sound for multimedia.

DAT and CD players both have the advantage that they can generate digital output, which can be read in by a suitably equipped computer without the need for extra digitizing hardware. In this respect, they resemble DV cameras. Where a digital signal cannot be produced, or where the computer is not fitted with the appropriate digital audio input, a digitizing sound card must be fitted to the computer, in the same way as a video capture board must be used for analogue video. Digital audio inputs are surprisingly uncommon, so it is often necessary for the (analogue) line output of a DAT or CD player to be redigitized by the sound card. This is clearly unfortunate, since it is preferable to work entirely with digital data and prevent noise and signal degradataion. It does, however, avoid the problem of incompatible sampling rates that can occur if, say, a recording on DAT is to be combined with an extract from a CD. Resampling audio is as undesirable as resampling images.

6
Which makes some assumptions about the auditory system's transient response, which may or may not be valid.

7
According to legend, the time to play Karajan's recording of Beethoven's 9th symphony.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 407
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

⇨ The necessity to resample data sampled at 48 kHz often occurs if the sound is to be combined with video. Some video applications do not yet support the higher sampling rate, even though DAT is widely used for capturing sound, and sound cards that support 48 kHz are becoming common. For multimedia work it may therefore be preferable to sample sound at 44.1 kHz, which is supported by all the major desktop video editing programs.

Sampling relies on highly accurate clock pulses to determine the intervals between samples. If the clock drifts, so will the intervals. Such timing variations are called *jitter*. The effect of jitter is to introduce noise into the reconstructed signal. At the high sampling frequencies required by sound, there is little tolerance for jitter: it has been estimated that for CD quality sound, the jitter in the ADC must be less than 200 picoseconds (200×10^{-12} seconds).

Even if they are inaudible, frequencies in excess of 20 kHz are present in the spectra of many sounds. If a sampling rate of around 40 kHz is used, these inaudible components will manifest themselves as aliasing when the signal is reconstructed. In order to avoid this, a filter is used to remove any frequencies higher than half the sampling rate before the signal is sampled.

Quantization

We mentioned in Chapter 2 that the number of quantization levels for analogue to digital conversion in any medium is usually chosen to fit into a convenient number of bits. For sound, the most common choice of sample size is 16 bits, as used for CD audio, giving 65 536 quantization levels. This is generally sufficient to eliminate quantization noise, if the signal is dithered, as we will describe shortly. As with images, smaller samples sizes (lower bit-depths, as we would say in the context of images) are sometimes needed to maintain small file sizes and bit rates. The minimum acceptable is 8-bit sound, and even this has audible quantization noise, so it can only be used for applications such as voice communication, where the distortion can be tolerated. In the search for higher fidelity reproduction, as many as 24 bits are sometimes used to record audio samples, but this imposes considerable demands on the accuracy of ADC circuitry.

Quantization noise will be worst for signals of small amplitude. In the extreme, when the amplitude is comparable to the difference

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 408
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

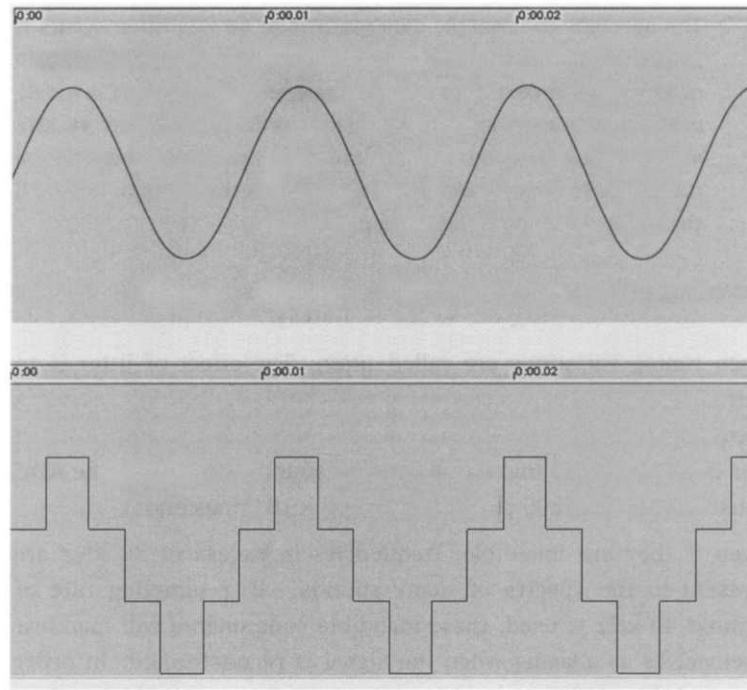


Figure 12.8
Undersampling a pure sine wave

between quantization levels, an analogue signal will be coarsely approximated by samples that jump between just a few quantized values. This is shown in Figure 12.8. The upper waveform is a pure sine wave; below it is a digitized version, where only four levels are available to accommodate the amplitude range of the original signal.[8] Evidently, the sampled waveform is a poor approximation of the original. The approximation could be improved by increasing the number of bits for each sample, but a more economical technique, resembling the anti-aliasing applied when rendering vector graphics, is usually employed. Its operation is somewhat counter-intuitive.

Before sampling, a small amount of random noise is added to the analogue signal. The word 'dithering' (which we used with a somewhat different meaning in Chapter 6) is used in the audio field to refer to this injection of noise. The effect on sampling is illustrated in Figure 12.9. The upper waveform is the original sine wave with added dither;[9] The lower waveform is a sampled version of this dithered signal. What has happened is that the presence of the noise has caused the samples to alternate rapidly between quantization levels, instead of jumping cleanly and abruptly from

8
If you want to be scrupulous, since these images were prepared using a digital audio application, the top waveform is a 16-bit sampled sine wave (a very good approximation), the lower is the same waveform downsampled to 2 bits.

9
We have used rather more noise than is normal, in order to show the effect more clearly.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 409
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

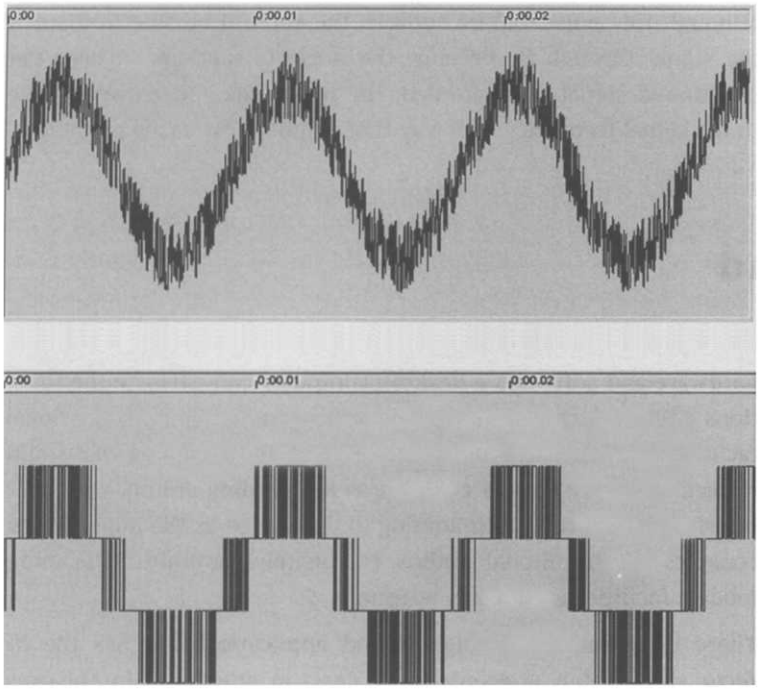


Figure 12.9
Dithering

one to the next, as they do in Figure 12.8. The sharp transitions have been softened. Putting it another way, the quantization error has been randomized. The price to be paid for the resulting improvement in sound quality is the additional random noise that has been introduced. This is, however, less intrusive than the quantization noise it has eliminated.

Plate 21 illustrates the effect of sampling and dithering on the signal's frequency spectrum. In these pictures, the horizontal x-axis represents frequency, the vertical y-axis amplitude, with the colours being used as an extra visual indication of intensity, and the back-to-front z-axis represents time. The first spectrum is the pure sine wave; as you would expect, it is a spike at the wave's frequency, which is constant over time. To its right is the spectrum of the sampled signal: spurious frequencies and noise have been introduced. These correspond to the frequency components of the sharp edges. Below the pure sine wave is the spectrum of the dithered version. The extra noise is randomly distributed across frequencies and over time. In the bottom left is the sampled version of this signal. The pure frequency has re-emerged clearly, but random noise is present where before there was none. However,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 410
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

although this noise will be audible, the ear will be able to discern the signal through it, because the noise is random. Where the undithered signal was sampled, the noise was concentrated near to the signal frequency, in a way that is much less easily ignored.

Processing Sound

With the addition of suitable audio input, output and processing hardware and software, a desktop computer can perform the functions of a modern multi-track recording studio. Such professional facilities are expensive and demanding on resources, as you would expect. They are also as complex as a recording studio, with user interfaces that are as intimidating to the novice as the huge mixing consoles of conventional studios. Fortunately, for multimedia, more modest facilities are usually adequate.

There is presently no single sound application that has the *de facto* status of a cross-platform desktop standard, in the way that Photoshop and Premiere, for example, do in their respective fields. Several different packages, some of which require special hardware support, are in use. Most of the well known ones are biased towards music, with integrated support for MIDI sequencing (see page 405) and multi-track recording. Several more modest programs provide simple recording and effects processing facilities; where hardware support is not provided, real-time effects are not usually achievable. Video editing packages usually include some integrated sound editing and processing facilities, and some offer basic sound recording. These facilities may be adequate for multimedia production in the absence of special sound software, and are especially convenient when the audio is intended as a soundtrack to accompany picture.

Given the absence of an industry standard sound application for desktop use, we will describe the facilities offered by sound programs in general terms only, without using any specific example.

Recording and Importing Sound

Many desktop computers are fitted with built-in microphones, and it is tempting to think that these are adequate for recording sounds.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 411
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

It is almost impossible to obtain satisfactory results with these, however — not only because the microphones themselves are usually small and cheap, but because they are inevitably close to the machine's fan and disk drives, which means that they will pick up noises from these components. It is much better to plug an external microphone into a sound card, but if possible, you should do the actual recording onto DAT (or good quality analogue tape) using a professional microphone, and capture it in a separate operation. Where sound quality is important, or for recording music to a high standard, it will be necessary to use a properly equipped studio. Although a computer can form the basis of a studio, it must be augmented with microphones and other equipment in a suitable acoustic environment, so it is not really practical for a multimedia producer to set up a studio for one-off recordings. It may be necessary to hire a professional studio, which offers the advantage that professional personnel will generally be available.

Before recording, it is necessary to select a sampling rate and sample size. Where the sound originates in analogue form, the choice will be determined by considerations of file size and bandwidth, which will depend on the final use to which the sound is to be put, and the facilities available for sound processing. As a general rule, the highest possible sampling rate and sample size should be used,[10] to minimize deterioration of the signal when it is processed. If a compromise must be made, the effect on quality of reducing the sample size is more drastic than that of reducing the sampling rate: the same reduction in size can be produced by halving the sampling rate or halving the sample size; the former is better. If the signal is originally a digital one — the digital output from a DAT recorder, for example — the sample size should be matched to the incoming rate, if possible.

A simple calculation suffices to show the size of digitized audio. The sampling rate *is* the number of samples generated each second, so if the rate is r Hz and the sample size is s bits, each second of digitized sound will occupy $rs/8$ bytes. Hence, for CD-quality, $r = 44.1\times10^3$ and $s = 16$, so each second occupies just over 86 kbytes,[11] each minute roughly 5 Mbytes. These calculations are based on a single channel, but audio is almost always recorded in stereo, so the estimates should be doubled. Conversely, where stereo effects are not required, the space occupied can be halved by recording in mono.

10 But remember that if the sound is to be combined with video in an editing application that cannot handle 48 kHz, then it is better to record at 44.1 kHz in the first place.

11 In kHz, k represents 1000, following normal usage, but in kbytes, the k is 1024, in accordance with the conventions of computing.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 412
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

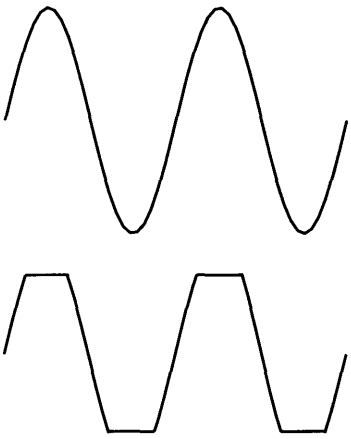


Figure 12.10
Clipping

Professional sound applications will record directly to disk, so that the possible length of recordings is limited only by the available disk space and any file size limitations built in to the operating system. Many lower-level programs record to RAM, however, and subsequently carry out all their processing in memory. While this is more efficient, it imposes severe restrictions on the length of sound that can be managed.

The most vexatious aspect of recording is getting the levels right. If the level of the incoming signal is too low, the resulting recording will be quiet, and more susceptible to noise; if the level is too high, clipping will occur, that is, at some points, the amplitude of the incoming signal will exceed the maximum value that can be recorded. The value of the corresponding sample will be set to the maximum, so the recorded waveform will apparently be clipped off straight at this threshold. (Figure 12.10 shows the effect on a pure sine wave.) The result is heard as a particularly unpleasant sort of distortion. Ideally, a signal should be recorded at the highest possible level that avoids clipping. Sound applications usually provide level meters, so that the level can be monitored, with clipping alerts. Where the sound card supports it, a gain control can be used to alter the level. Some sound cards do not provide this function, so that the only option is to adjust the output level of the equipment from which the signal originates. Setting the level correctly is easier said than done, especially where live recordings are being made: to preserve the dynamic range of the recording, the same gain must be used throughout, but the optimum can only be determined at the loudest point. When the sound is live, this cannot be known in advance, and only experience can be used to choose gain settings. Where the material already exists on tape or CD, it is possible — and usually necessary — to make several passes in order to find the best values.

Some software includes automatic gain controls, which vary the gain dynamically according to the amplitude of the signal, in order to prevent clipping. They must, therefore, reduce the volume of louder passages, so as a side-effect, they reduce the dynamic range of the recording. This is generally undesirable, but may be necessary if suitable levels cannot be maintained throughout the recording.

⇨ It may be obvious, but it seems worth emphasizing: once a signal has been clipped, nothing can be done to restore it. Reducing the amplitude subsequently just produces a smaller clipped signal. There is no way to recover the lost waveform. Similarly, although

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 413
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

sound programs often provide a facility for 'normalizing' a sound after recording, by amplifying it as much as possible without causing clipping, this stretches the dynamic range of the original without adding any more detail. In practice it may be necessary to use this facility, or to select and amplify particularly quiet passages within a sound editing application after the recording has been made. In principle, though, the gain should always be set correctly, both when recording to tape, and when recording or capturing to disk.

A technically simpler alternative to recording sound is to import it from an audio CD. Although audio CDs use a different format from CD-ROM, they nevertheless are a structured collection of digital data, so they can be read by suitable software. QuickTime includes an audio CD import component that allows any sound application based on QuickTime to open tracks on a CD just like any other file. This is the simplest way of importing sounds, but most recorded music is copyrighted, so it is necessary to obtain permissions first. Copyright-free collections can be obtained, much like royalty-free image libraries, although they tend to the anodyne. Composers and musicians with access to professional recording facilities may supply their work on CD, avoiding the need for the multimedia producer to deal with the sound recording process. However, even when importing sounds from CDs there can be difficulty in getting the levels right.

The Internet is a rich source of ready-made sounds, but many are made available illegally, and others may not be legally reproduced without payment of a fee. Increasingly, though, record companies are arriving at mechanisms to provide music online, usually in the form of MP3 files (see below). While it may be legal to download these files and listen to them, it remains generally illegal to use them in any published form without obtaining clearance from the copyright holders.

Sound Editing and Effects

We can identify several classes of operation that we might want to apply to recorded sounds. Most of them have counterparts in video editing, and are performed for similar reasons.

First, there is editing, in the sense of trimming, combining and rearranging clips. The essentially time-based nature of sound

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 414
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

naturally lends itself to an editing interface based on a timeline. A typical sound editing window is divided into tracks, in imitation of the separate tape tracks used on traditional recording equipment, providing a clear graphic representation of the sound through time. The sound in each track may usually be displayed as a waveform; the time and amplitude axes can be scaled, allowing the sound to be examined in varying degrees of detail. Editing is done by cutting and pasting, or dragging and dropping, selected parts of the track. Each stereo recording will occupy two tracks, one for each channel. During the editing process many tracks may be used to combine sounds from separate recordings. Subsequently, these will be mixed down onto one or two tracks, for the final mono or stereo output. When mixing, the relative levels of each of the tracks can be adjusted to produce the desired balance — between different instruments, for example.

A special type of edit has become common in audio: the creation of loops. Very short loops are needed to create voices for the electronic musical instruments known as samplers (whose functions are increasingly performed by software). Here, the idea is to create a section of sound that represents the sustained tone of an instrument, such as a guitar, so that arbitrarily long notes can be produced by interpolating copies of the section between a sample of the instrument's attack and one of its decay. It is vital that the sustained sample loops cleanly; there must not be abrupt discontinuities between its end and start, otherwise audible clicks will occur where the copies fit together. Although some software makes such loops automatically, using built-in heuristics such as choosing zero crossings for each end of the loop, the best results require a detailed examination of the waveform by a person. Longer loops are used in certain styles of dance music — techno and drum'n'bass, for example — which are based on the combination of repeating sections. Again, there is a requirement for clean looping, but this time, at the coarser level of rhythmic continuity.

As well as editing, audio has its equivalent of post-production: altering sounds to correct defects, enhance quality, or otherwise modify their character. Just as image correction is described in terms of filters, which are a digital equivalent of traditional optical devices, so sound alteration is described in terms of gates and filters, by analogy with the established technology. Whereas analogue gates and filters are based on circuitry whose response produces a desired effect, digital processing is performed by

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 415
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

algorithmic manipulation of the samples making up the signal. The range of effects, and the degree of control over them, that can be achieved in this way is much greater than is possible with analogue circuits. Several standard plug-in formats are in use, that allow effects to be shared among programs. Although it is not an audio application, Premiere's effects plug-in format is becoming widely used; at a more professional level, the formats associated with Cubase VST and with DigiDesign ProTools are popular.

The most frequently required correction is the removal of unwanted noise. For example, in Figure 12.1, it might be considered desirable to remove the background noises that were unavoidably picked up by the microphone, since the recording was made in the open. A *noise gate* is a blunt instrument that is used for this purpose. It eliminates all samples whose value falls below a specified threshold; samples above the threshold are left alone. As well as specifying the threshold, it is usual to specify a minimum time that must elapse before a sequence of low amplitude samples counts as a silence, and a similar limit before a sequence whose values exceed the threshold counts as sound. This prevents the gate being turned on or off by transient glitches. By setting the threshold just above the maximum value of the background noise, the gaps between words will become entirely silent. Since the noise gate has no effect on the speaker's words, the accompanying background noise will cut in and out as he speaks, which may well turn out to be more distracting than the original noise. This illustrates a general problem with noise removal: the noise is intimately combined with the signal, and although people can discriminate between the two, computer programs generally cannot.

Noise gates can be effective at removing hiss from music, since, in this case, the noise is hidden except in silent passages, where it will be removed by the noise gate. There are more sophisticated ways of reducing noise than the all-or-nothing filtering of the noise gate, though. Filters that remove certain bands of frequencies can be applied to noise that falls within a specific frequency range. *Low pass* filters, which allow low frequencies to pass through them, removing high frequencies, can be used to take out hiss; *high pass filters*, which pass the high frequencies and block the low, are used to remove 'rumble': low frequency noise caused by mechanical vibrations. A *notch filter* removes a single narrow frequency band. The commonest use of notch filters is to remove hum picked up from the mains: this will have a frequency of exactly 50 or 60 Hz,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 416
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

depending on the part of the world in which the noise was recorded. Some sophisticated programs offer the user the ultimate facility of being able to redraw the waveform, rubbing out the spikes that correspond to clicks, and so on. To do this effectively, however, requires considerable experience and the ability to interpret the visual display of a waveform in acoustic terms, which, as the examples shown earlier demonstrate, is not always easy.

Specialized filters are available for dealing with certain common recording defects. A *de-esser* is a filter that is intended to remove the sibilance that results from speaking or singing into a microphone placed too close to the performer. *Click repairers* are intended to remove clicks from recordings taken from damaged or dirty vinyl records.[12] Although these filters are more discriminating than a noise gate, they are not infallible. The only sure way to get perfect sound is to start with a perfect take — microphones should be positioned to avoid sibilance, and kept well away from fans and disk drives, cables should be screened to avoid picking up hum, and so on.

12
There are also effects plug-ins that attempt to add authentic-sounding vinyl noise to digital recordings.

⇨ Although the noise reduction facilities available in desktop sound applications are fairly crude and ineffectual, more elaborate — and computationally expensive — approaches have been developed. One approach is based on attempting to analyse the acoustic properties of the original recording apparatus on the basis of the make-up of the noise in quiet passages, and then compensating for it in the music. Sophisticated noise reduction techniques are used to restore old records from the early part of the 20th century, and also to reconstruct other damaged recordings, such as the tapes from voice recorders of crashed aircraft.

When we consider effects that alter the quality of a sound, there is a continuum from those that perform minor embellishments to compensate for poor performance and recording, to those that radically alter the sound, or create new sounds out of the original. A single effect may be used in different ways, at different points in this continuum, depending on the values of parameters that affect its operation. For example, a *reverb* effect is produced digitally by adding copies of a signal, delayed in time and attenuated, to the original. These copies model reflections from surrounding surfaces, with the delay corresponding to the size of the enclosing space and the degree of attenuation modelling surfaces with different acoustic reflectivity. By using small delays and low reflectivity, a recording can be made to sound as if it had been made inside a small

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 417
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

room. This degree of reverb is often a necessary enhancement when the output from electric instruments has been recorded directly without going through a speaker and microphone. Although cleaner recordings are produced this way, they are often too dry acoustically to sound convincing. Longer reverb times can produce the illusion of a concert hall or a stadium.[13] Still longer times, with the delayed signals being amplified instead of attenuated, can be used creatively to generate sustained rhythm patterns from a single chord or note.

13
One sound application offers the values 'Empty room', 'Concert Hall', 'Stadium', and 'Outer Space' as settings for its reverb effect.

Other effects can be put to a variety of uses in a similar way. These include *graphic equalization*, which transforms the spectrum of a sound using a bank of filters, each controlled by its own slider, and each affecting a fairly narrow band of frequencies. (Analogue graphic equalizers are commonly found on mid-range domestic sound systems.) These can be used to compensate for recording equippment with idiosyncratic frequency response, or to artificially enhance the bass, for example, to produce a desired frequency balance. *Envelope shaping* operations change the outline of a waveform. The most general envelope shapers allow the user to draw a new envelope around the waveform, altering its attack and decay and introducing arbitrary fluctuations of amplitude. Specialized versions of envelope shaping include *faders*, which allow a sound's volume to be gradually increased or decreased, and *tremolo*, which causes the amplitude to oscillate periodically from zero to its maximum value.[14]

14
To classical musicians, 'tremolo' means the rapid repetition of a single note — this does produce a periodic oscillation of amplitude. The 'tremolo arm' fitted to Fender Stratocasters and other electric guitars actually produces a periodic change of pitch, more accurately referred to as 'vibrato'.

Time stretching and *pitch alteration* are two closely related effects that are especially well-suited to digital sound. With analogue recordings, altering the duration of a sound can only be achieved by altering the speed at which it is played back, and this alters the pitch. With digital sound, the duration can be changed without altering the pitch, by inserting or removing samples. Conversely, the pitch can be altered without affecting the duration.

Time stretching is most often required when sound is being synchronized to video or another sound. If, for example, a voice-over is slightly too long to fit over the scene it describes, the soundtrack can be shrunk in time, without raising the pitch of the speaker's voice, which would happen if the voice track was simply played at a faster speed.

Pitch alteration can be used in several ways. It can be applied uniformly to alter the pitch of an instrument, compensating for an out-of-tune guitar, for example. It can be applied periodically to add a vibrato (periodic fluctuation of pitch) to a voice or instrument, or

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 418
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

it can be applied gradually, to produce a 'bent note', in the same way a blues guitarist changes the tone of a note by bending the string while it sounds. The all-important shape of the bend can be specified by drawing a curve showing how the pitch changes over time.

Beyond these effects lie what are euphemistically called 'creative' sound effects. Effects such as flanging, phasing, chorus, ring modulation, reversal, Doppler shift, and wah-wah, which were pioneered in the 1960s on albums such as *Sergeant Pepper's Lonely Hearts Club Band* and *Electric Ladyland* have been reproduced digitally, and joined by new extreme effects such as roboticization. These effects, if used judiciously, can enhance a recording, but they are easily over-used, and are generally best enjoyed in private.

Compression

While the data rate for CD-quality audio is nothing like as demanding as that for video, it still exceeds the bandwidth of dial-up Internet connections, and lengthy recordings rapidly consume disk space. A single three minute song, recorded in stereo, will occupy over 25 Mbytes. Hence, where audio is used in multimedia, and especially when it is delivered over the Internet, there is a need for compression. The complex and unpredictable nature of sound waveforms makes them difficult to compress using lossless methods. Huffman coding can be effective in cases where the amplitude of the sound mainly falls below the maximum level that can be represented in the sample size being used. In that case, the signal could have been represented in a smaller sample size, and the Huffman algorithm, by assigning short codes to the values it does encounter, will effectively do this automatically. This is a special case, though, and, in general, some form of lossy compression will be required.

An obvious compression technique that can be applied to speech is the removal of silence. That is, instead of using 44,100 samples with the value of zero for each second of silence (assuming a 44.1 kHz sampling rate) we record the length of the silence. This technique appears to be a special case of run-length encoding, which, as we said in Chapter 5, is lossless. However, as Figure 12.1 shows, 'silence' is rarely absolute. We would obtain little compression if we

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 419
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

simply run-length encoded samples whose value was exactly zero; instead, we must treat samples falling below a threshold as if they were zero. The effect of doing this is equivalent to applying a noise gate, and is not strictly lossless, since the decompressed signal will not be identical to the original.

The principles behind lossy audio compression are different from those used in lossy image compression, because of the differences in the way we perceive the two media. In particular, whereas the high frequencies associated with rapid changes of colour in an image can safely be discarded, the high frequencies associated with rapid changes of sound are highly significant, so some other principle must be used to decide what data can be discarded.

Speech Compression

Telephone companies have been using digital audio since the early 1960s, and have been forced by the limited bandwidth of telephone lines to develop compression techniques that can be effectively applied to speech. An important contribution of this early work is the technique known as *companding*. The idea is to use non-linear quantization levels, with the higher levels spaced further apart than the low ones, so that quiet sounds are represented in greater detail than louder ones. This matches the way in which we perceive differences in volume.

Figure 12.11 shows an example of non-linear quantization. The signal value required to produce an increase of one in the quantized value goes up logarithmically. This produces compression, because fewer bits are needed to represent the full range of possible input values than a linear quantization scheme would require. When the signal is reconstructed an inverse process of expansion is required, hence the name 'companding' — itself a compressed version of 'compressing/expanding'.

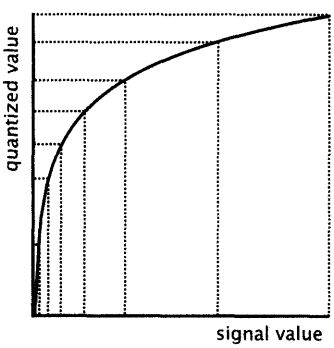


Figure 12.11
Non-linear quantization

Different non-linear companding functions can be used. The principle important ones are defined by ITU Recommendations for use in telecommunications. Recommendation G.711 defines a function called the *μ-law*, which is used in North America and Japan. It has been adopted for compressing audio on Sun and NeXT systems, and files compressed in accordance with it are commonly found on the Internet. A different ITU Recommendation is used in the rest of the world, based on a function known as the *A-law*.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 420
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Telephone signals are usually sampled at only 8 kHz. At this rate, μ-law compression is able to squeeze a dynamic range of 12 bits into just 8 bits, giving a one-third reduction in data rate.

The μ-law is defined by the equation:

$$y = \frac{\log(1 + \mu x)}{\log(1 + \mu)} \text{ for } x \geq 0$$

where μ is a parameter that determines the amount of companding; $\mu = 255$ is used for telephony.

The A-law is:

$$y = \begin{cases} \frac{Ax}{1 + \log A} \text{ for } 0 \leq |x| \leq 1/A \\ \frac{1 + \log Ax}{1 + \log A} \text{ for } 1/A \leq |x| \leq 1 \end{cases}$$

Another important technique that was originally developed for, and is widely used in, the telecommunications industry is *Adaptive Differential Pulse Code Modulation (ADPCM).* This is related to inter-frame compression of video, in that it is based on storing the difference between consecutive samples, instead of the absolute value of each sample. Because of the different nature of audio and video, and its origins in hardware encoding of transmitted signals, ADPCM works somewhat less straightforwardly than a simple scheme based on the difference between samples.

Storing differences will only produce compression if the differences can be stored in fewer bits than the sample. Audio waveforms can change rapidly, so, unlike consecutive video frames, there is no reason to assume that the difference will necessarily be much less than the value. Basic *Differential Pulse Code Modulation (DPCM)*[15] therefore computes a predicted value for a sample, based on preceding samples, and stores the difference between the prediction and the actual value. If the prediction is good, the difference will be small. *Adaptive* DPCM obtains further compression by dynamically varying the step size used to represent the quantized differences. Large differences are quantized using large steps, small differences using small steps, so the amount of detail that is preserved scales with the size of the difference. The details of how this is done are complicated, but as with companding, the effect is to make efficient use of bits to store information, taking account of its rate of change.

ITU Recommendation G.721 specifies a form of ADPCM representation for use in telephony, with data rates of 16 kbps and 32 kbps.

15 Pulse Code Modulation is the term used in audio and communications circles for encoding digital data as a sequence of pulses representing ones and zeroes. Whereas this is more or less the only sensible representation for computer use, alternatives, such as Pulse Width Modulation, exist where the data is to be represented as a stream for transmission, rather than as stored values.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 421
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Lower rates can be obtained by a much more radical approach to compression. *Linear Predictive Coding* uses a mathematical model of the state of the vocal tract as its representation of speech. Instead of transmitting the speech as audio samples, it sends parameters describing the corresponding state of the model. At the receiving end, these parameters can be used to construct the speech, by applying them to the model. The details of the model and how the parameters are derived from the speech lie beyond the scope of this book. Speech compressed in this way can be transmitted at speeds as low as 2.4 kbps. Because the sound is reconstructed algorithmically, it has a machine-like quality, so it is only suitable for applications where the content of the speech is more important than a faithful rendition of someone's voice.

Perceptually Based Compression

The secret of effective lossy compression is to identify data that doesn't matter — in the sense of not affecting perception of the signal — and throw it away. If an audio signal is digitized in a straightforward way, data corresponding to sounds that are inaudible may be included in the digitized version. This is because the signal records all the physical variations in air pressure that cause sound, but the perception of sound is a sensation produced in the brain, via the ear, and the ear and brain do not respond to the sound waves in a simple way.

Two phenomena in particular cause some sounds not to be heard, despite being physically present. Both are familiar experiences: a sound may be too quiet to be heard, or it may be obscured by some other sound. Neither phenomenon is quite as straightforward as it might appear.

The *threshold of hearing* is the minimum level at which a sound can be heard. It varies non-linearly with frequency, as shown in Figure 12.12. A very low or very high frequency sound must be much louder than a mid-range tone to be heard. It is surely no coincidence that we are most sensitive to sounds in the frequency range that corresponds to human speech. When compressing sound, there is no point in retaining sounds that fall below the threshold of hearing, so a compression algorithm can discard the corresponding data. To do this, the algorithm must use a *psycho-acoustical model* — a mathematical description of aspects of the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 422
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

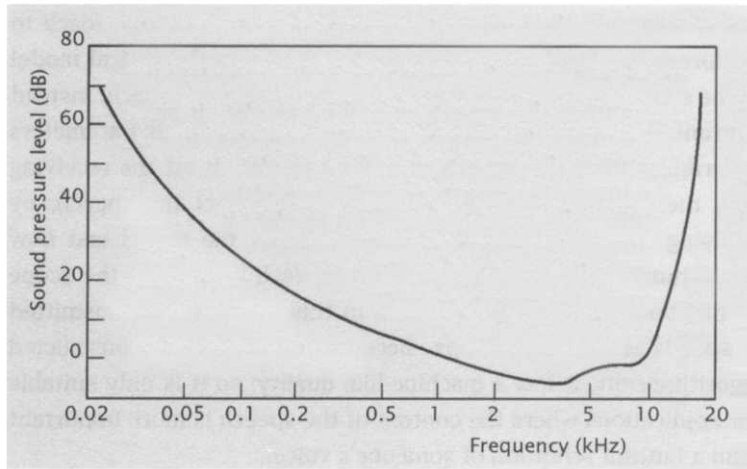


Figure 12.12
The threshold of hearing

way the ear and brain perceive sounds. In this case, what is needed is a description of the way the threshold of hearing varies with frequency.

Loud tones can obscure softer tones that occur at the same time.[16] This is not simply a case of the loud tone 'drowning out' the softer one; the effect is more complex, and depends on the relative frequencies of the two tones. *Masking*, as this phenomenon is known, can be conveniently described as a modification of the threshold of hearing curve in the region of a loud tone. As Figure 12.13 shows, the threshold is raised in the neighburhood of the masking tone. The raised portion, or *masking curve* is non-linear, and assymmetrical, rising faster than it falls. Any sound that lies within the masking curve will be inaudible, even though it rises above the unmodified threshold of hearing. Thus, there is an additional opportunity to discard data. Masking can be used more cleverly, though. Because masking hides noise as well as some components of the signal, quantization noise can be masked. Where a masking sound is present, the signal can be quantized relatively coarsely, using fewer bits than would otherwise be needed, because the resulting quantization noise can be hidden under the masking curve.

It is evident that the phenomena just described offer the potential for additional compression. It is not obvious how a compression algorithm can be implemented to take advantage of this potential. The approach usually adopted is to use a bank of filters to split the signal into bands of frequencies; thirty two bands are commonly

16
In fact, they can also obscure softer tones that occur a little later or, strange as it may seem, slightly earlier.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 423
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

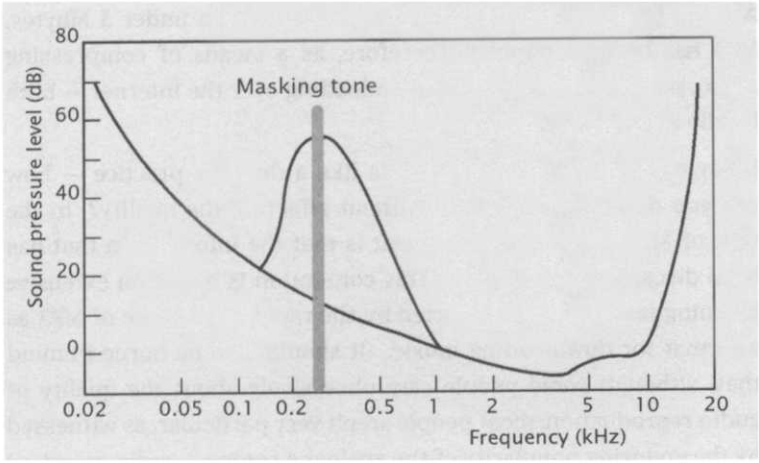


Figure 12.13
Masking

used. The average signal level in each band is calculated, and using these values and a psycho-acoustical model, a masking level for each band is computed. That is, it is assumed that the masking curve within each band can be approximated by a single value. If the signal in a band falls entirely below its masking level, that band is discarded. Otherwise, the signal is quantized using the least number of bits that causes the quantization noise to be masked.

Turning the preceding sketch into a working algorithm involves many technical details that lie beyond the scope of this book. The best known algorithms that have been developed are those specified for audio compression in the MPEG standards. MPEG-1 and MPEG-2 are primarily video standards, but, since most video has sound associated with it, they also include audio compression. MPEG audio has been so successful that it is often used on its own purely for compressing sound, especially music.

MPEG-1 specifies three *layers* of audio compression. All three layers are based on the principles just outlined. The encoding process increases in complexity from Layer 1 to Layer 3, while as a result, the data rate of the compressed audio decreases, from 192 kbps for each channel at Layer 1, to 128 kbps at Layer 2, and 64 kbps at Layer 3. (These data rates will be doubled for stereo.) The audio part of the MPEG-2 standard is essentially identically with MPEG-1 audio, except for some extensions to cope with surround sound.

MPEG-1 Layer 3 audio, or *MP3* as it is usually called,[17] achieves compression ratios of around 10:1, while maintaining high quality.

17
MP3 does not, despite what you will sometimes read, stand for MPEG-3. There is no MPEG-3.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 424
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

A typical track from a CD can be compressed to under 3 Mbytes. MP3 has become popular, therefore, as a means of compressing audio, particularly music, for downloading over the Internet — both legally and illegally.

Lossy compression always sounds like a dubious practice — how can you discard information without affecting the quality? In the case of MPEG audio, the argument is that the information that has been discarded is inaudible. This contention is based on extensive listening tests, and is supported by the rapid acceptance of MP3 as a format for downloading music. (It should also be borne in mind that, although some people care obsessively about the quality of audio reproduction, most people aren't very particular, as witnessed by the enduring popularity of the analogue compact audio cassette.) As with any lossy form of compression, though, MPEG audio will deteriorate progressively if it is decompressed and recompressed a number of times. It is therefore only suitable as a delivery format, and should not be used during production, when uncompressed audio should be used whenever possible.

Formats

Most of the development of digital audio has taken place in the recording and broadcast industries, where the emphasis is on physical data representations on media such as compact disc and digital audio tape, and on data streams for transmission and playback. There are standards in these areas that are widely adhered to. The use of digital sound on computers is a much less thoroughly regulated area, where a wide range of incompatible proprietary formats and *ad hoc* standards can be found. The standardizing influence of the Internet is, as yet, less pronounced in audio than it is in graphics.

File Formats

Each of the three major platforms has its own sound file format: AIFF for MacOS, WAV (or WAVE) for Windows, and AU[18] for Unix. All three file types have evolved over the years until they now provide broadly similar capabilities. Each can store audio data at a variety of commonly used sampling rates and sample sizes, including the CD and DAT standard values. Each supports uncompressed or

18 More properly, WAV is the Audio Waveform file format, actually a variety of Microsoft RIFF file, and AU is the NeXT/Sun audio file format.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 425
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

compressed data, with a range of compressors, including μ-law and A-law (although not, at the time of writing, MP3). They are all, to some extent, extensible formats, which will be able to accommodate new compressors and sampling rates in the future. But each one has its own idiosyncrasies; in particular, AU files impose some constraints on the combinations of compressor, sampling rate and sample size allowed.

MP3 has its own file format, which does not accommodate sound compressed by any other method.

As well as pure audio formats, video and other multimedia formats can accommodate sound as one media type among several. QuickTime and AVI files usually include one or more audio tracks in addition to video, and they are sometimes used purely for audio. SWF (Flash) files can also be used to store sound.

Although no standard audio file format has emerged, support for AIFF, WAV, and AU files is common on all platforms. In particular, QuickTime — and, therefore, any program that uses it — is capable of reading and writing sound in all three of these formats, as well as MP3 and some other formats associated with particular sound applications.

Streaming Audio

In Chapter 10, we explained that streamed video resembles broadcast television. Streamed audio resembles broadcast radio. That is, sound is delivered over a network and played as it arrives, without having to be stored on the user's machine first. As with video, this allows live transmission and the playing of files that are too big to be held on an average-sized hard disk. Because of the lower bandwidth required by audio, streaming is more successful for sound than it is for video.

The first succesful streaming audio format was Real Networks' RealAudio, a companion to RealVideo, which has now been incorporated into Real G2. This uses a proprietary compressor to maintain reasonable sound quality at data rates suitable for dial-up Internet connections. Streaming QuickTime can also be used for audio, on its own as well as accompanying video. Both formats are used for broadcasting live concerts and for the Internet equivalent of radio stations, and also for providing 'play on demand' music — the Internet as the biggest jukebox in the world.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 426
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

MP3 requires too much bandwidth to be streamed over modems, although it is well within the capacity of cable modems and ADSL. A 'lo-fi' MP3 format is in use, however, which sacrifices some quality to bandwidth, allowing MP3 audio to be streamed at lower data rates.

⇨ Another streamed audio format has a somewhat different emphasis. Liquid Audio is intended as an architecture for distributing music. Liquid Audio can be streamed, in order to preview a song, for example, but the primary intention is for the user then to download a CD-quality file, which can then be burned on to a recordable CD. Mechanisms for payment and the prevention of piracy are built in to the architecture.

MIDI

If we had written a piece of music, there are two ways we could send it to you. We could play it, record the performance, and send you the recording, or we could write it down using some form of notation, indicating the arrangement, and send you the sheet music, so you could play the piece for yourself. There is a parallel here with bitmapped and vector graphics. In the first case, we send you the actual sound, in the second, we send you what amounts to a set of instructions telling you how to produce the sound. In either case we are making some assumptions about what you can do: for the recording, we assume you have a machine capable of playing back whichever medium we have recorded our performance on. For the sheet music, we are making the more demanding assumptions that you can read our chosen music notation, have access to the instrument or instruments indicated in the arrangement, and can either play yourself or can get musicians to play those instruments. If the music is arranged for a symphony orchestra, this may present some difficulties for you, whereas, if we were to send a recording, all the difficulties would lie at our end.

In the digital realm, there is a similar choice of options for delivering music. So far, we have considered ways of delivering digitized sound, that is, the equivalent of recordings. There also exists an equivalent to delivering the sheet music, i.e. a way of delivering instructions about how to produce the music, that can be interpreted by suitable software or hardware. Similar assumptions must be made: for sound files, you must have software that can read them — as we have seen, this is not a demanding requirement.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 427
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

For instructions, you must have software that can interpret the instructions, and some means of producing sounds that correspond to the appropriate instruments.

MIDI (The Musical Instruments Digital Interface) provides a basis for satisfying these requirements. Originally, MIDI was devised as a standard protocol for communicating between electronic instruments, such as synthesizers, samplers, and drum machines. By defining a standard hardware interface, and a set of instructions indicating such things as the start and end of a note, it provided a means of controlling a collection of such instruments from a single keyboard, removing the requirement for the huge banks of keyboards beloved of certain 1970s rock stars, and opening the way for playing traditional keyboard instruments, particularly synthesizers, with other controllers, such as drum pads or wind instruments.

More significantly, perhaps, MIDI allowed instruments to be controlled automatically by devices that could be programmed to send out sequences of MIDI instructions. Originally, *sequencers*, as these devices are known, were dedicated hardware devices, programmed using their own built-in, relatively clumsy, interfaces. It was not long before it was realized that computer programs could offer a more convenient and flexible means of sequencing, provided that a computer could be fitted with a MIDI interface so that it could send the necessary signals to other MIDI devices. Such an interface is a relatively simple and inexpensive device, so computer-based sequencers rapidly became available. A software sequencer provides editing and compositional functions, so it needs to store MIDI sequences in files. This requirement led to the development of a standard file format for MIDI files, that is, a way of storing MIDI on disk. Clearly, such files can be exchanged between computers equipped with MIDI software. They can also be incorporated into multimedia productions.

Playing back MIDI files requires an instrument that understands MIDI, but a computer, equipped with suitable hardware or software, can be such an instrument itself. Sounds can either be synthesized on a sound card, or held on disk in the form of samples, to be played back in response to MIDI instructions. MIDI files are therefore a means of communicating music. Because they do not contain any audio data, they can be much more compact than actual digitized sound files. For the same reason, though, they cannot guarantee the same fidelity: the samples available when the file is produced

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 428
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

may be of higher quality than those used to play it back — just as the musician who plays a piece of music from a score may not be sufficiently accomplished to realise the composer's intentions. In both cases, the result is unpredictable.

MIDI Messages

A MIDI *message* is an instruction that controls some aspect of the performance of an instrument. Messages are encoded in much the same way as machine instructions: a *status byte* indicates the type of the message, and is followed by one or two *data bytes* giving the values of parameters. Although wind instruments, drum pads, and guitars are used as MIDI controllers (as devices that transmit MIDI signals are called), MIDI is markedly biased towards keyboard instruments. Thus, for example, the most commonly used message is 'Note On', which takes two parameters: a number between 0 and 127 indicating the note to be sounded, and a key velocity, indicating how fast the key was pressed, and hence the attack of the note. When an actual keyboard is being used to generate MIDI messages, these values will be sensed by the keyboard's hardware as the musician plays the key. When the message is being generated by software, the values are specified by the user.

Other notable MIDI messages include 'Note Off', which ends a note, 'Key Pressure', which indicates the degree of 'aftertouch' to be applied, and 'Pitch Bend', to change note values dynamically, as a guitarist does by bending the string (on MIDI keyboards, a wheel is used for this function).

The status and data bytes in a stream of MIDI instructions are distinguishable by the value of their most significant bit. This makes possible an optimization: where a sequence of messages all have the same status byte, it may be omitted from the second and subsequent messages, for which it will be inferred from the most recent value. This arrangement is called *running status*; it can save an appreciable number of bytes where a sequence of notes is being played with no modifications. Using the convention that the end of a note can be indicated by a 'Note On' message with a velocity of zero, the whole sequence can consist of a single 'Note On' status byte, followed by a series of data bytes giving the notes to be played and the velocities to be applied to them.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 429
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

When a MIDI message is interpreted, we say that an event occurs. In live performance, the timing of events is determined by the player in real time. In a MIDI file, it is necessary to record the time of each event. Each message is preceded by a *delta time*, that is, a measure of the time since the preceding event. Near the beginning of the file is a specification of the units to be used for times.

General MIDI and the QuickTime Music Architecture

The preceding account indicates how notes are produced, but leaves unanswered the question of how they are to be associated with particular sounds. Typically, the sorts of instruments controlled by MIDI — synthesizers and samplers — provide a variety of *voices*. In the case of synthesizers, these are different synthesized sounds;[19] in the case of samplers, different instrument samples. A MIDI 'Program Change' message selects a new voice, using a value between 0 and 127. The mapping from these values to voices is not specified in the MIDI standard, and may depend on the particular instrument being controlled. There is thus a possibility that a MIDI file intended to specify a piece for piano and violin might end up being played on trombone and kettle drum. To help overcome this unsatisfactory situation, an addendum to the MIDI standard, known as *General MIDI*, was produced, which specifies 128 standard voices to correspond to the values used by Program Change messages. The assignments are shown in Table 12.1.

19 Often called *patches* by synthesists.

For drum machines and percussion samplers, Program Change values are interpreted differently as elements of drum kits — cymbals of various sorts, snares, tom-toms, and so on (see Table 12.2).

General MIDI only associates program numbers with voice *names*. There is no guarantee that identical sounds will be generated for each name by different instruments — a cheap sound card may attempt to synthesize all of them, while a good sampler may use high quality samples of the corresponding real instruments. Adherence to General MIDI offers some guarantee of consistency, though, which is otherwise entirely missing.

QuickTime incorporates MIDI-like functionality. QuickTime Musical Instruments provide a set of instrument samples[20], and the QuickTime Music Architecture incorporates a superset of the features of MIDI. QuickTime can read standard MIDI files, so any computer with QuickTime installed can play MIDI music using software alone.

20 The Roland Sound Canvas set is distributed with QuickTime.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 430
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Table 12.1
General MIDI voice numbers

1	Acoustic Grand Piano	44	Contrabass	87	Synth Lead 7
2	Bright Acoustic Piano	45	Tremolo Strings	88	Synth Lead 8
3	Electric Grand Piano	46	Pizzicato Strings	89	Synth Pad 1
4	Honky-tonk Piano	47	Orchestral Harp	90	Synth Pad 2
5	Rhodes Piano	48	Timpani	91	Synth Pad 3
6	Chorused Piano	49	Acoustic String Ensemble 1	92	Synth Pad 4
7	Harpsichord	50	Acoustic String Ensemble 2	93	Synth Pad 5
8	Clavinet	51	Synth Strings 1	94	Synth Pad 6
9	Celesta	52	Synth Strings 2	95	Synth Pad 7
10	Glockenspiel	53	Aah Choir	96	Synth Pad 8
11	Music Box	54	Ooh Choir	97	Ice Rain
12	Vibraphone	55	Synvox	98	Soundtracks
13	Marimba	56	Orchestra Hit	99	Crystal
14	Xylophone	57	Trumpet	100	Atmosphere
15	Tubular bells	58	Trombone	101	Bright
16	Dulcimer	59	Tuba	102	Goblin
17	Draw Organ	60	Muted Trumpet	103	Echoes
18	Percussive Organ	61	French Horn	104	Space
19	Rock Organ	62	Brass Section	105	Sitar
20	Church Organ	63	Synth Brass 1	106	Banjo
21	Reed Organ	64	Synth Brass 2	107	Shamisen
22	Accordion	65	Soprano Sax	108	Koto
23	Harmonica	66	Alto Sax	109	Kalimba
24	Tango Accordion	67	Tenor Sax	110	Bagpipe
25	Acoustic Nylon Guitar	68	Baritone Sax	111	Fiddle
26	Acoustic Steel Guitar	69	Oboe	112	Shanai
27	Electric Jazz Guitar	70	English Horn	113	Tinkle bell
28	Electric clean Guitar	71	Bassoon	114	Agogo
29	Electric Guitar muted	72	Clarinet	115	Steel Drums
30	Overdriven Guitar	73	Piccolo	116	Woodblock
31	Distortion Guitar	74	Flute	117	Taiko Drum
32	Guitar Harmonics	75	Recorder	118	Melodic Tom
33	Wood Bass	76	Pan Flute	119	Synth Tom
34	Electric Bass Fingered	77	Bottle blow	120	Reverse Cymbal
35	Electric Bass Picked	78	Shakuhachi	121	Guitar Fret Noise
36	Fretless Bass	79	Whistle	122	Breath Noise
37	Slap Bass 1	80	Ocarina	123	Seashore
38	Slap Bass 2	81	Square Lead	124	Bird Tweet
39	Synth Bass 1	82	Saw Lead	125	Telephone Ring
40	Synth Bass 2	83	Calliope	126	Helicopter
41	Violin	84	Chiffer	127	Applause
42	Viola	85	Synth Lead 5	128	Gunshot
43	Cello	86	Synth Lead 6		

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 431
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

QuickTime can also control external MIDI devices. MIDI tracks can be combined with audio, video or any of the other media types supported by QuickTime.

MIDI Software

MIDI sequencing programs, such as Cakewalk Metro and Cubase, perform capture and editing functions equivalent to those of video editing software. They support multiple tracks, which can be allocated to different voices, thus allowing polytimbral music to be constructed. In addition, such packages support composition.

Music can be captured as it is played from MIDI controllers attached to a computer via a MIDI interface. The sequencer can generate metronome ticks to assist the player to maintain an accurate tempo. Although it is common to simply use the sequencer as if it were a tape recorder, to capture a performance in real time, sometimes MIDI data is entered one note at a time, which allows musicians to 'play' music that would otherwise be beyond their competence. Facilities normally found in conventional audio recording software are also available, in particular, the ability to 'punch in': the start and end point of a defective passage are marked, the sequencer starts playing before the beginning, and then switches to record mode, allowing a new version of the passage to be recorded to replace the original.

Sequencers will optionally *quantize* tempo during recording, fitting the length of notes to exact sixteenth notes, or eighth note triplets, or whatever duration is specified. This allows rhythmically loose playing to be brought into strict tempo, which may be felt desirable for certain styles of music, although the result often has a machine-like feel, since live musicians rarely play exactly to the beat. Quantization is usually necessary if it is desired to produce a transcription of the music, since otherwise the program will transcribe exactly what was played, even if that involves dotted sixty-fourth notes and rests.

Most programs allow music to be entered using classical music notation, often by dragging and dropping notes and other symbols onto a stave. Some programs allow printed sheet music to be scanned, and will perform optical character recognition to transform the music into MIDI. The opposite transformation, from MIDI to a printed score, is also often provided, enabling transcriptions

Table 12.2
General MIDI Drum Kit Numbers

35	Acoustic Bass Drum
36	Bass Drum 1
37	Side Stick
38	Acoustic Snare
39	Hand Clap
40	Electric Snare
41	Lo Floor Tom
42	Closed Hi Hat
43	Hi Floor Tom
44	Pedal Hi Hat
45	Lo Tom Tom
46	Open Hi Hat
47	Low -Mid Tom Tom
48	Hi Mid Tom Tom
49	Crash Cymbal 1
50	Hi Tom Tom
51	Ride Cymbal 1
52	Chinese Cymbal
53	Ride Bell
54	Tambourine
55	Splash Cymbal
56	Cowbell
57	Crash Cymbal 2
58	Vibraslap
59	Ride Cymbal 2
60	Hi Bongo
61	Low Bongo
62	Mute Hi Conga
63	Open Hi Conga
64	Low Conga
65	Hi Timbale
66	Lo Timbale

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 432
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

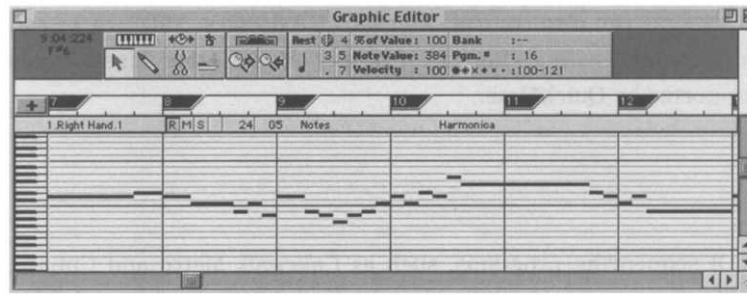


Figure 12.14
The 'piano-roll' interface to a sequencer

of performed music to be made automatically. Those who do not read music usually prefer to use the 'piano-roll' interface, illustrated in Figure 12.14, which allows the duration of notes to be specified graphically, essentially using a timeline. Their pitch is specified with reference to the piano keyboard shown on the left. For music constructed out of repeating sections, loops can be defined and reused many times.

Once a piece of music has been recorded or entered, it can be edited: individual notes' pitch and duration can be altered, sections can be cut and pasted, or global changes can be made, such as transposing the entire piece into a different key, or changing the time signature. The parameters of individual MIDI events can be changed — the velocity of a note can be altered, for example. Voices can be changed to assign different instruments to the parts of the arrangement.

Because digital audio is very demanding of computer resources, but MIDI is much less so, the two forms of music representation were originally separated, with different software being used for each. Now that computers have become powerful enough to take audio in their stride, the two are commonly integrated in a single application, which allows MIDI tracks to be combined and synchronized with full audio. This arrangement overcomes one of the major limitations of MIDI, namely the impossibility of representing vocals (except for 'Oohs' and 'Aahs'). MIDI can be transformed into audio, much as vector graphics can be rasterized and transformed into pixels. The reverse transformation is sometimes supported, too, although it is more difficult to implement. MIDI captures the musical structure of sound, since MIDI events correspond to notes. Being able to transform audio into MIDI allows music to be recorded from ordinary instruments instead of MIDI controllers — it can even be recorded from somebody's whistling — and then edited or transcribed in terms of musical notes.[21]

21
It is sometimes claimed that facilities of this sort allow people who cannot play instruments to compose tunes by whistling. You have to be able to whistle in tune, though.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 433
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

⇨ The *MOD file* is a file format which is being increasingly used for music. Originating on the Amiga platform, MOD (short for 'module') files combine the features of audio and MIDI files. In particular, they can include the necessary samples to use to play back a piece of music whose structure is described by MIDI information. Programs for playing MOD files are called *trackers*.

Combining Sound and Picture

Sound is frequently used as a part of a video or animation production. When it is, synchronization between sound and picture becomes a matter of considerable importance. This is seen most clearly where the picture shows a person talking, and the soundtrack contains their speech. If synchronization is slightly out, the result will be disconcerting; if it is considerably out, the result will at best be unintentionally funny, but more likely, incoherent. Although speech makes the most exacting demands on synchronization, wherever sound and picture are related, it is necessary that the temporal relationship between them is maintained. Voice-overs should match the picture they describe, music will often be related to edits, and natural sounds will be associated with events on screen.

In order to establish synchronization, it is necessary to be able to identify specific points in time. Film is divided into frames, which provide a natural means of identifying times. Video tape does not have physical frames, but, as we mentioned in Chapter 10, timecode can be written to a video tape, allowing the frames to be identified. Audio tape can be similarly augmented with timecode, and the codes can be used to synchronize tape recorders, both audio and video. As long as the timecodes on two machines agree, they must be in synch.

Sound is effectively continuous, though, even in the digital domain — the high sampling rates used for digital sound mean that a single sample defines too short a time interval to be useful. For sound, the division into frames imposed by timecode is just a useful fiction. This fictional division continues to be used when synching digital audio and video. It enables sound and picture tracks in a video editing application such as Premiere to be arranged on the same timeline. Unlike the soundtrack on a piece of film or a video tape,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 434
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

a sound track in Premiere is physically independent of the video it accompanies, so it is easy to move the sound in time relative to the picture, simply by sliding the sound track along the timeline. This is not something you would normally want to do if the sound and picture had originally been recorded together. In that case, you will usually want to maintain their synchronization during editing. For this purpose, tracks can be locked together, so that, for example, cutting out part of the video track will remove the accompanying sound.

Audio tracks may be displayed as waveforms. When a sound track has been made independently of the picture — a voice-over or musical accompaniment, for example — it will be necessary to fit the sound to the picture. By looking at the waveform to identify the start of syllables in speech, or stressed beats in music, an editor can identify meaningful points in the sound track, which can be lined up with appropriate picture frames. Performing this matching by eye is difficult, so a method that is often used is to scrub through the sound to identify the precise cue point by ear, and place a marker (a facility offered by Premiere, for example) which can then be lined up on the timeline with the video frame (which can also be marked for identificiation). Sometimes, it may be necessary to apply a time-stretching filter to adjust the duration of the sound to fit the picture, as described earlier.

Synchronization can thus be established in a video editing program, but it must then be maintained when the video and its sound track are played back — possibly over a network. If the sound and video are physically independent — travelling over separate network connections, for example — synchronization will sometimes be lost. This is a fact of life and cannot be avoided. Audio and video data streams must therefore carry the equivalent of timecode, so that their synchronization can be checked, and they can be resynched, if necessary. Usually, this will require some video frames to be dropped, so that picture can catch up with sound — the greater data rate of the video means that it is the more likely to fall behind.

Where video and audio are played back from a local hard disk, it is easier to maintain synchronization, although it still cannot be guaranteed, owing to the different speeds of the components involved. For very short clips, it is possible to load the entire sound track into memory before playback begins, eliminating any potential delays caused by reading from disk. This is impractical for movies of any significant duration. For these, it is normal to *interleave* the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 435
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

audio and video, that is, the audio is divided into chunks, which are interspersed between video frames.[22] The size of chunk can be varied, and optimized for particular hardware configurations.

22 AVI stands for 'audio and video interleaved', even though by default they aren't.

➲ QuickTime provides highly flexible support for specifying time within a movie. A movie has a *time base*, specifying the current time and the rate (and direction) at which time passes, relative to a *time coordinate system*, which specifies the units in which time is to be measured. The time coordinate system is chosen to suit the data; for example, sound sampled at 44.1 kHz may use a time coordinate in which each time unit is 1/44.1 ms. A *clock component* is used to obtain real time values, usually from the clock chip built in to the computer on which the movie is being played. By synchronizing the time base to the clock, the required playback rate is maintained. Whenever some data, such as a frame of video, is to be played, the time base is passed to the software component responsible for playing the data. The current movie time can then be used to determine the appropriate portion of the data to play. If sound and picture start to drift, this will ensure that they are brought back into the correct temporal relationship.

Further Information

[Poh95] is an excellent introduction to all aspects of digital audio, and [Rot92] is a good, brief account of MIDI. [RM94] describes the techniques of sound recording.

Exercises

1. Give an example of a natural sound that has an identifiable pitch.
2. Why are the sampling frequencies normally used for 'lo-fi' digital sound exact sub-multiples of 44.1 kHz?
3. Given that singing has characteristics of both speech and music, which compression algorithms would you expect to be most successful on songs?
4. A problem commonly encountered when recording in the open air is that a microphone will pick up the sounds made

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 436
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

by the wind blowing against it. Describe how you would attempt to remove such noise from a digitized recording. How successful would you expect your attempts to be? Suggest an alternative approach to eliminating wind noise.

5. When we described anti-aliasing of vector graphics in Chapter 4, it was as an antidote to insufficiently high sampling rates. In this chapter, we have described dithering as a way of mitigating the effect of insufficient quantization levels. Would this kind of dithering help improve the apparent quality of under-sampled audio? Is there any connection between dithering sound and dithering colour, as described in Chapter 6?

6. Explain how you would apply a 'cross-fade' to a stereo recording of a solo instrument, so that the sound would appear to move gradually from the extreme left to the extreme right of the sound stage.

7. What could you do to correct a sound that was digitized with its levels (a) too high; (b) too low. How would you prepare a sound recording with an extremely wide dynamic range for a multimedia production?

8. Is there a limit on how far you can (a) stretch, (b) contract a digitized sound successfully? What aspects of particular kinds of sound might affect the limits?

9. (a) Describe the alterations that must be made to the digital representation of a sound to raise its pitch by an octave, without changing its duration. (Raising the pitch by an octave is the same as doubling the frequency.)

   (b) Describe the alteration that must be made to a MIDI Note-On message to raise the note's pitch by an octave.

10. Under what circumstances might you expect to lose synchronization between sound and picture in a multimedia production? What steps could you take to minimize the chances of this happening.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 437
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Combining Media

13

Broadly speaking, there are two models currently in use for combining elements of different media types: page-based and synchronization-based.

In the first, text, images, and video are laid out in a two-dimensional arrangement that resembles the way text and images are laid out in books and magazines. Time-based elements, such as video clips and sound, are embedded in the page as if they were images, occupying a fixed area; controls may be provided to start and stop playback. Individual pages can be combined using the linking mechanism of hypertext. Such linked page-based multimedia productions are known as *hypermedia*. The best known example of a hypermedia system is the World Wide Web.

Hypermedia takes its inspiration from paper-based media that are essentially static, and grafts time-based features onto a spatially organized framework. In contrast, synchronization-based multimedia makes time the central organizing principle. Elements are arranged in time, often using a timeline, so that they are presented as a sequence, resembling a slide show. *Transitions*, such as dissolves and wipes, may be used to go from one element in the sequence to the next. Unlike most (but not all) slide shows, however, a

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 438
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

multimedia *presentation* (as such timeline-based productions are usually known) will often incorporate parallelism: several video clips may be shown at the same time, perhaps overlaid against a static image, or a sound track may play during an animation. Elements may be synchronized, so that, for example, some text may be displayed as long as a video clip is playing, or an image will be displayed ten seconds after the beginning of a video clip.

⇨ A third model for combining media is exemplified by VRML, as we described briefly in Chapter 11. Here, the elements are laid out in a three-dimensional scene. The user can move about in the 3-D space and inspect images or objects that they encounter. Links may be incorporated, to provide hypertext functionality, and interactivity can be provided, using the scripting mechanisms that are applied to the other models of multimedia, as we will describe in Chapter 14. Scene-based multimedia of this sort is presently most often found in games, although these are not usually thought of as multimedia productions. The apparent failure of VRML so far to achieve wide acceptance means that currently there is no standard form for scene-based multimedia. The proposed MPEG-4 standard for multimedia, presently under development, uses a scene-based model, but has, as yet, only been implemented experimentally.

Multimedia productions based on both of these models are often augmented with interactive features that allow the user to control their progress. As we will describe in Chapter 14, a scripting facility is usually provided that allows multimedia authors to write simple programs that cause actions to occur in response to user input. Scripts can also be written to cause actions to occur after a certain period of time, or in response to events such as a video clip finishing. By using scripts of this sort, temporal organization can be added to page-based multimedia. Non-linear branching can be provided in a synchronized presentation, either using scripting or by extending the timeline metaphor to support linking. Since it is also the case that the elements of a synchronization-based multimedia presentation must be arranged spatially, the distinction between our models of media combination is somewhat blurred. Nevertheless, their different emphases make it useful to consider them, at least to begin with, as distinct ways of combining media.

⇨ More than one introduction to multimedia, when describing the combination of media into multimedia, does so in a chapter entitled 'Putting it all together'. Aside from the regrettable descent into cliché, this title carries the suggestion that it is somehow necessary,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 439
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

once you have moved beyond a single medium, to combine the whole lot: text, images, video, animation, and sound. It isn't, and may often be undesirable to do so.

We can illustrate this using familiar pre-digital media. Laurence Bergreen has written an excellent biography of Louis Armstrong.[1] However, as it is a book, it offers no opportunity to hear the music, which is something of a deficiency, given its subject. All the important recordings are available, but obtaining them all, collating them with the text, and actually getting round to listening to them, requires a certain amount of effort, discipline and money. Surely a case for multimedia.

1 Laurence Bergreen, *Louis Armstrong: An Extravagant Life*, Harper Collins, 1997.

Such was presumably the thought in the minds of the producers of the television documentary *The Wonderful World of Louis Armstrong*, which broadly covers the same material as Bergreen's book. The multimedia possibilities of television allowed them to combine extracts from Armstrong's recordings with spoken commentary and reminiscences from those who knew him. However, the documentary makers apparently had little access to any relevant footage or images of the trumpeter, and since they evidently believe in the dictum that television is a visual medium, they were left with the quandary of what to do when there are no appropriate pictures. They chose to use computer graphics and post-production techniques to produce 'evocative' imagery and altered video to accompany the words and music. The result was unwatchable, but not untypical of much contemporary television.

The trouble is that working in a visual medium demands images, whether appropriate ones are available or not. The forms of digital multimedia offer the opportunity to select and combine those media that are suitable to a particular production, without necessarily putting it *all* together.

Hypermedia

Hypertext, as we described it in Chapter 8, never generated much excitement. It required the addition of first graphics and then other media elements to the linking mechanism of hypertext to produce something that captured people's imaginations: *hypermedia*. It is easy to see how this generalization can be achieved by considering HTML and the World Wide Web, though you should realize from the history in Chapter 9, that the Web was by no means the first hypermedia system.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 440
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

HTML and Hypermedia

In HTML, links are implemented as anchors, with an `href` attribute whose value is a URL pointing to the destination. In previous sections, we assumed that the URL points to a page of HTML. What if it doesn't?

HTTP doesn't care. If you ask a server to return the resource identified by a URL it will do so. As we mentioned in Chapter 2, it will include in its response, as well as the resource's data, an indication of what sort of data it is, in the form of a MIME content type. The presence of MIME content types in HTTP responses answers a question that may have bothered you: how does a Web browser know that what it is receiving is an HTML document? Because the content type is specified as `text/html`. The question begs another: How does an HTTP server know what MIME type to specify when it sends a response? It would be unrealistic to expect the server to analyze the data and deduce its type; a more prosaic solution is adopted. The server has access to a configuration database, maintained by whoever looks after the server, which provides a mapping from filename extensions, or file type and creator codes, to MIME types.

The upshot of all this is that when a browser receives a document it also receives information about the type of data it contains, in the form of a MIME content type. This brings us back to the original question, which we can now rephrase: What does the browser do if the content type is not `text/html`? Either the browser will be intelligent enough to deal with it anyway, or it will have to get another program to do so.

Consider the second option first. Web browsers can be configured so that, for each MIME type, a program is nominated to deal with documents of that type. The nominated programs are usually called *helper applications*. When a document arrives that the browser cannot display itself, it starts up the appropriate helper application to do so instead. For example, suppose (on a Macintosh) your browser had been configured so that the video player Sparkle was the helper application for data of type `video/mpeg` (MPEG video clips). If you click on an anchor whose `href` attribute's URL points to an MPEG video clip, the data will be retrieved as usual. When it arrives, the browser will use the OS's facilities to start up another process to run Sparkle, handing it the retrieved data. The video

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 441
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

will be displayed in a new window belonging to Sparkle, with that program's interface, independent of the Web browser. Thus, although you can play the video clip after retrieving it via a Web page, it is not integrated with any other Web data — we have not really achieved hypermedia, although we can link together different media using hyperlinks.

In order to properly integrate the display of media besides formatted text into a Web browser we must extend the capabilities of the browser so that it can render other media, and we must provide some extra HTML tags to control the layout of pages with embedded graphics, video and sound. It is not realistic to expect a Web browser to be able to cope with absolutely any imaginable type of data; for some obscure types a helper application will always be the best option. On the other hand, some types, especially image types and plain text, are so common that it is reasonable to expect browsers to have code to display them built in. Other types, such as video and audio fall in between: it would be nice to handle them in the browser, but they are not yet so common that the necessary implementation effort on the part of the browser manufacturer is justified, and not all users will be happy with the resulting increased resource requirements of their browsers. The solution to this dilemna is *plug-ins*: software modules that can be installed independently by users who need them. Plug-ins are loaded by a browser when it starts up and add functionality to it, specifically the ability to deal with additional media types. The major browsers all use the same interface that allows them to incorporate plug-ins, and they can be configured to use a plug-in instead of a helper application for certain media types. An example of a widely used plug-in is Apple's QuickTime plug-in, which transparently incorporates all the functionality of QuickTime into any browser that implements the Netscape plug-in interface (this includes Internet Explorer). It is written by the QuickTime engineers, who presumably know more about QuickTime than anyone else, so it provides a better implementation of QuickTime displaying than is likely to have been provided if the browsers had been extended by their own developers. Users only need to install the plug-in (usually simply by placing it in an appropriate place) if they want to view QuickTime within their browsers; this way, users with no interest in video do not incur the associated overhead.

Once a browser becomes capable of rendering non-textual data without the aid of a helper, the possibility exists of integrating the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 442
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

display of such data with the other elements of Web pages. This leads to a mode of multimedia presentation based on a *page layout* model. This model is derived from long-established practice in print-based media for combining text and graphics. It can be fairly naturally extended to incorporate video and animation, by treating them as if they were pictures to be placed on the page, but with the added property that they can be made to move. Sound sits much less comfortably on a page, being purely non-visual. An expedient often employed is to represent a sound by an icon or set of controls, which can be treated as a graphic element, and then activated to play the sound in the same way as a video element is activated. An alternative is simply to associate a sound with a page, and have it play when the page is loaded. This approach has the disadvantage that users have no control over whether the sound is played.

Special markup is required for embedding these new elements in Web pages. In HTML 4.0, the `OBJECT` element is provided for embedded media of all types — it is flexible enough to accommodate new media types that have not yet been implemented, as well as embedded executable content, in particular, Java applets. Previous versions of HTML only provided support for bitmapped images, in the form of the `IMG` element. Use of this element has become so well established that, for the most popular image formats — JPEG, GIF, and possibly PNG — it is unlikely to be superseded by `OBJECT`. Before HTML 4.0 there was no officially endorsed method for embedding any other type of media, but Netscape implemented an `EMBED` element (subsequently also adoped by Microsoft in Internet Explorer) for including video, sound, and other media types. Although `EMBED` never had any official status, it has been widely used, and is routinely generated by HTML authoring programs.

Part of the enduring attraction of `IMG` is its simplicity. It is an empty element; in the simplest case, it has a single attribute, `src`, whose value is a URL pointing to an image file.[2] `IMG` is an inline element, so the image is displayed where the tag occurs. This enables you to run images in with text, use them as headings or labels on list items, and so on; alternatively, you can isolate an image by enclosing it in its own paragraph.

2 Strictly, in HTML 4, one other attribute, `alt`, is compulsory — see below.

To understand the necessity for `IMG`, consider the following two fragments of HTML.

```
Let us show you a <A href="still1.jpg">picture</A>.
```

and

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 443
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

```
This picture has more presence:
<P>
<IMG src="still1.jpg">
</P>
```

Let us show you a picture.

This picture has more presence:



Figure 13.1
Linked and embedded images

Figure 13.1 shows the corresponding parts of the displayed page. In the first case, the word 'picture' is highlighted as a link, just as if it pointed to another page. When the highlighted text is clicked, the image is displayed alone in the browser window, replacing the current page (or frame). In the second case, the image is displayed as part of the page, like a picture in a newspaper or magazine.

Such clumsy layout would not cut much ice in the magazine design world, though, nor on today's World Wide Web. Like any other HTML document element, an `IMG` can be laid out in a more imaginative and pleasing way using CSS rules. Margins and borders can be set around images, which can be aligned and floated, using the properties introduced in Chapter 8. For complete layout control, absolute positioning, including z-index, can be used to put images exactly where you want them — subject always to the proviso that not all user agents interpret stylesheet information. (For the benefit of older browsers, `IMG` elements have a set of attributes, including `height`, `width`, `border` and `align` to provide some degree of control over their placement.)

`OBJECT` elements are the preferred way of embedding multimedia and executable content in Web pages.[3] We will not consider the latter at this stage, but we note that there are additional attributes for this purpose. For embedding images, video, sound, and so on, the `data` attribute is used; the value is a URL that points to the data to be rendered as the object's content. It is advisable also to specify the MIME type of the data, as the value of the `type` attribute. Although the server will supply this information if the data is requested, providing it in the object's start tag means that a user agent that cannot render data of that type need not waste time downloading it — possibly a non-trivial consideration if the object is a large video clip.

The way in which `OBJECT` provides for the possibility of a user agent's being unable to display the specified object is ingenious, but slightly counter-intuitive. Unlike an `IMG` element, an `OBJECT` has content, but the content is not the object — that is specified by the `data` attribute. The content is displayed only if the user agent is unable to display the object. This means that the content can be

3 Preferred, that is, by the W3C. At the time of writing, the implementation of `OBJECT` in contemporary (fourth generation) browsers is patchy, but `EMBED` is almost universally supported.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 444
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

used to provide a replacement version of the object. The following use is typical:

```
<P>Here's a groovy moovy
<P>
<OBJECT data="movies/clip2.mov"
        type="video/quicktime">
  <OBJECT data="images/still2.jpg" type="image/jpeg">
    A 5 second video clip.
  </OBJECT>
</OBJECT>
```

If possible, a movie clip is displayed. If the necessary plug-in is not there, or some other reason prevents playing the movie, a still image is put in its place. If even this cannot be done, some text describing the missing object is used.

It is often necessary or desirable to specify some aspects of the way in which an object is displayed. For example, a video clip might be shown in a standard video controller component, with play, pause and fast-forward controls, or it might be displayed with no controls and play itself automatically. A page designer would want to be able to specify which of these styles is used. However, different options make sense for different media: you cannot sensibly attach a video controller component to a still image, for example. Furthermore, OBJECT elements are intended to be able to support arbitrary media, including those not yet invented. Consequently, it is not possible to define a set of attributes for the OBJECT element type that will provide for all possible eventualities.[4] Instead, parameters required to control the display of objects are set using a special element type: PARAM. This is similar to META, in that it has attributes name and value, which are used to provide a value for a named parameter. The parameters for an object are set in PARAM elements contained in the OBJECT element. The precise set of names that is applicable depends on the type of the object's data (or, on the plug-in used to display the object).

For example, the QuickTime plug-in understands several parameters that can have the value true or false, including controller, which determines whether a movie is displayed with a standard movie controller component, autoplay, which causes the movie to start playing as soon as the page is displayed, if it is true, and loop, which makes it play as a continuous loop. Hence, to make a movie play automatically forever without any visible controls, the following HTML markup could be used.

4 This argument has extra force when you consider that OBJECTs can be used to contain applets, which may need to be supplied with parameters when they are run. The set of parameters required will be determined by the applet's programmer, and cannot be predicted.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 445
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

```
<OBJECT data="movies/clip2.mov"
        type="video/quicktime">
  <PARAM name = "controller" value = "false">
  <PARAM name = "autoplay" value = "true">
  <PARAM name = "loop" value = "true">
</OBJECT>
```

⇨ The EMBED element type has attributes with the same names as these parameters.

Links

One of the places you can use images is between the start and end tags of an A element with an **href** attribute. The effect is to produce a clickable image that serves as the source of a link. This is usually indicated by outlining it in a special colour. A common use of this facility is to create clickable icons or buttons; another is to produce image catalogues consisting of small 'thumbnail' pictures:[5] clicking on a thumbnail causes a full-sized version of the picture to be displayed. Alternatively, it may just be considered more appropriate to have an image serve as the linking element, if it somehow captures the semantics of what it links to. For example, an image of a book's cover may be used as the source of a link to a page of an on-line book store from which it can be purchased. (It is, however, more difficult to convey a link's meaning unambiguously using an image than it is using words.)

5 Web graphics applications often provide a means of constructing such catalogues automatically, just by selecting a directory of image files.

⇨ You can use objects as link sources in the same way, but for media other than images it is not clear that this is a useful thing to do. In particular, for time-based media, there is something counter-intuitive about considering the whole object as a link. It seems more natural to use events or particular frames in this way. This raises many questions, since to do this requires us to incorporate time into the hitherto simple, and essentially spatial, model of nodes and links. We will defer consideration of how to go about doing so until the second half of this chapter.

A slightly more elaborate way of using images as links is extremely popular: an *image map* is an image containing 'hot' areas, which are associated with URLs. Clicking on such an area causes the resource identified by its associated URL to be retrieved. Figure 13.2 shows such an image map: it looks like just a picture, but each of the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 446
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

flower heads is the source of a link. To make an image into an image map, it must be associated with a MAP element, by giving it a usemap attribute whose value is a fragment identifier. This must match the value of the name attribute of a MAP. Thus, for example,

```
<IMG src="flower1.jpeg" usemap="#image-map-1">
```

associates the image flower1.jpeg with a map whose start tag looks like:

```
<MAP name="image-map-1">
```

The content of a MAP is a series of AREA elements, each having an href attribute whose value is a URL in the usual way, together with two attributes shape and coords, which together describe a region within the image which is to be linked to the specified URL. The options for shape and the corresponding interpretation of coords are listed in Table 13.1. The MAP element associated with the flower image looks like this:



Figure 13.2
An image map

```
<MAP name="image-map-1">
<AREA shape="rect" coords="65,351,166,413"
      href="section1.html">
<AREA shape="rect" coords="64,353,167,410"
      href="section1.html">
<AREA shape="rect" coords="106,324,124,442"
      href="section1.html">
<AREA shape="rect" coords="351,420,452,482"
      href="section2.html">
 eleven more AREA elements
 </MAP>
```

(As you can see, AREA is an empty element type.) Each flower head is approximated by three intersecting rectangles.

Table 13.1
Shapes and co-ordinates

shape	coords
rect	*left-x, top-y, right-x, bottom-y*
circle	*center-x, center-y, radius*
poly	$x_1, y_1, x_2, y_2, \ldots, x_N, y_N$

Almost nobody will want to measure or calculate the coordinates of areas such as these. Fortunately, many tools are available that enable you to construct a map by selecting an area and typing a URL. Most graphics packages are also capable of constructing image maps. The particular example just shown was made in Illustrator, where it suffices to select an object and type a URL in a text field in the attributes palette. If you then export the image as a GIF or JPEG file you are given the option of having an HTML file containing the image map generated at the same time. Note, however, that whatever the shape of the object you select, only rectangles, circles and polygons can be used as areas in the HTML code.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 447
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Synchronization-Based Presentations

Pure hypermedia possesses no temporal structure. The way in which a hypermedia production unfolds in time is determined by the user's choice of links to follow or controls to activate. This is not always a desirable situation. For example, one might want to produce a multimedia demo showing off a computer system's sound, graphics and video capabilities that can be left running unattended at an exhibition or in a shop. For such applications, the order in which the elements of the production are played must be determined beforehand by the author. A popular framework for organizing multimedia in time is the timeline.

Timelines for multimedia authoring work in much the same way as the video editing and animation timelines described in Chapters 10 and 11. The sophistication with which they can be used varies widely. At the most basic end of the scale, simple slide show packages, such as Power Show, allow video clips and images to be sequenced by dragging them onto a 'filmstrip', which is simply a representation of the clips and images in the order they are to be displayed. The duration of each can be set, and transitions — wipes, dissolves, ripples, page turns, and so on — can be applied between them. Text can be superimposed onto the images, and animated, and sound can be associated with each element of the presentation. Although all of this could be achieved with a video editor such as Premiere or post-production package such as After Effects, slide show programs are simpler to use, cheaper to buy, and need not render their output as a movie; instead, they can just produce a set of instructions for playing the various items in the slide show in the designated order.

Timeline-based multimedia authoring can be much more complex, as demonstrated by Macromedia Director. In Director, a collection of media objects is known as a 'cast', with the individual objects being 'cast members'. A Director movie, as the finished production is called, is constructed by arranging cast members in time on a timeline, and in space. More precisely, since cast members can be used many times, pointers to them, known as 'sprites'[6] are arranged to make the movie. The timeline becomes the 'score', which shows, for each frame of the movie, all of the sprites that are active. The spatial arrangement of sprites is performed on the 'stage', which is simply an editable version of the display. Sprites can be animated

6 The terminology betrays Director's origins as an animation program.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 448
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

using key frames, as in Flash. To construct a movie, the author drags sprites onto the score, and specifies the number of frames for which they are active (usually by dragging over the score) or sets key frames and specifies tweening parameters.

The playing of a Director movie is described in terms of a playback head travelling through the score one frame at a time, at a rate prescribed by the author.[7] When the playback head is in a particular frame, all the sprites that are shown in the score at the corresponding point are displayed at the appropriate position — either where they have been placed on the stage, or at coordinates interpolated from their positions at nearby key frames. After the correct frame time has elapsed, the playback head moves on to the next frame, whose sprites are displayed. A sprite can refer to a sound or video clip, or even another Director movie, and these have their own natural duration. To ensure that they are played in their entirety (assuming that that is what is desired) they can either be extended over a sufficient number of frames, or the sprite can be placed into a single frame, and using a facility provided for the purpose, the playback head can be made to remain in that frame for as long as it takes the entire clip to play.

7
Or as near to that rate as the user's computer can manage.

Sprite properties such as transparency can be animated as well as their position, so Director can produce motion graphics effects in the same way as a program like After Effects. Transitions can also be used in a similar way to Premiere and other video editing applications. With all these possibilities, the timeline, in the guise of Director's score, can be a vehicle for producing elaborate presentations using all the media types we have considered. The facilities we have described only permit linear presentations; the non-linearity that characterizes hypermedia is missing. It can be provided by the use of additional facilities, which we will describe in Chapter 14.

SMIL

The timeline is a visual metaphor that makes it easy to organize multimedia presentations in time, but it is not the only way that such organization can be expressed. *SMIL (Synchronized Multimedia Integration Language)* is a tag-based language for specifying the temporal structure of a presentation, in a similar way to that in which the structure of a hypermedia document can be specified in HTML. Like HTML, SMIL, being a purely text-based language,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 449
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

can be written using any text editor, it doesn't require special authoring tools (although, again like HTML, elaborate tools can be used to generate SMIL code from a more comfortable environment — for example, a timeline). For our purposes, SMIL has the additional advantage that its textual representation lays bare some of the details of synchronization that are concealed by the timeline metaphor, and does so in a way that provides a direct comparison with the page-based hypermedia facilities of HTML.

The SMIL 1.0 specification was adopted as a W^{3}C Recommendation in June 1998. As this suggests, it is intended that SMIL presentations be played over the Internet, using URLs to identify individual media resources stored on a server. The widely used RealPlayer G2, although generally thought of as a program for watching streamed video over the Internet, is actually a SMIL player, and is capable of showing synchronized clips in parallel and sequentially. Similarly, QuickTime 4.1 incorporates SMIL.

SMIL is defined by an XML DTD, so the syntax of its tags will be familiar. At the outermost level, the document structure resembles that of an HTML document: the entire document is a `smil` element, whose content comprises a `head` followed by a `body`. Note that the element names, and thus their start and end tags are written in lower-case; unlike HTML, XML is case sensitive. The `head` may contain `meta` elements, which, again as in HTML, provide an open-ended mechanism for including in the document information about the document. The `head` may also include layout information, describing the spatial disposition of the elements of the presentation on the screen. This takes the form of a `layout` element, which contains definitions written either in CSS2, using its absolute positioning facilities, or in a simpler notation called the *SMIL Basic Layout Language*.[8] In either case, it is conventional (in SMIL Basic it is necessary) to define the location and dimensions of a collection of *regions*, each associated with one or more of the elements that occur in the document body. These have a `region` attribute identifying which of the defined regions should be used to display the element. We will not describe the syntactical details here — consult the SMIL specification if you need to know about the SMIL Basic Layout Language — and in the examples that follow we have elided references to layout and regions to avoid cluttering the presentation.

As you would expect, the real substance of a SMIL document is its `body`. The most interesting elements that may appear in the

8 SMIL allows for the possibility of providing alternative layouts in different languages. See the specification for details.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 450
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

body's content are the *synchronization elements*, for which temporal properties can be specified. These include two compound elements, `par` (short for 'parallel', and nothing to do with paragraphs) and `seq` (sequence). Each may contain *media object elements*, which specify the actual images, video clips, sound, and so on, that are used in the presentation (the sprites, in Director's terms). To accommodate complex synchronization relationships, compound synchronization elements may be nested. Elements that occur within a `par` element may overlap in time, whereas those within a `seq` are displayed in sequence. A simple example will illustrate how this works, and serve to introduce the attributes that SMIL uses to specify timing and synchronization.

Suppose we wish to assemble four elements for a presentation: two images, a QuickTime movie, and a sound clip, and that we want to present them so that the movie starts to play at the beginning of the presentation, and plays in its entirety. Initially, the first image is displayed together with the movie, but after five seconds it is replaced by the other image, which is displayed for ten seconds before the first image comes back for another fifteen seconds. While this is going on, the sound should start playing after five seconds, and continue until twenty seconds into the presentation. (You can safely assume that the layout defined for this presentation causes the various components to be arranged in a sensible way on the screen.) We have three things going on in parallel here: a movie, images, and sound, with the images themselves being displayed in a sequence, so the structure of the SMIL document's body will be a `par` element containing media objects for the movie and sound and a `seq` element containing the sequence of images. The complete body is as follows:[9]

9
Indentation has been used to emphasise the structure; it has no syntactical significance.

```
<body>
  <par>
    <video src="movies/m1.mov"
           type="video/quicktime"/>
    <seq>
      <img src="images/image1.jpeg"
           type="image/jpeg"
           dur="5s"/>
      <img src="images/image2.jpeg"
           type="image/jpeg"
           dur="10s"/>
      <img src="images/image1.jpeg"
           type="image/jpeg"
```

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 451
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

```
        dur="15s"/>
    </seq>
    <audio src="sounds/sound1"
        type="audio/aiff"
        begin="5s" end="20s"/>
  </par>
</body>
```

The media object elements are all empty;[10] they all use a `src` attribute to provide a URL specifying the whereabouts of the media data. (Here we have used relative URLs.) The media object elements allowed are `animation`, `audio`, `img`, `ref`, `text`, `textstream`, and `video`. The intention of most of these should be clear. The `ref` element is a catch-all for media that are not accurately described by any of the other tags. In fact, the other tags are purely descriptive synonyms for `ref`: the type of data is determined by the `type` attribute, which specifies a MIME type, or by information provided by the server.

10 Since this is XML, their tags end with />.

The first element in the `par` thus specifies the movie. The next is a `seq` for the sequenced images. We must use three `img` elements, even though two of them relate to the same image — each element corresponds to an occurrence of the image during a particular period of time. The elements within a `seq` are displayed in the order they appear textually, as you would probably expect. The duration of each can be specified using the `dur` attribute, which takes a SMIL *clock value*. Clock values can either be specified as a number of hours, minutes, seconds (as in our example) or milliseconds, using the abbreviations `h`, `m`, `s`, and `ms`, or in the form *hours:mins:secs.fraction*, which is much like a timecode, except that a fraction is used instead of a number of frames, since, in general, we cannot rely on any particular frame rate. The *hours:* and *.fraction* parts are optional. A special value `indefinite` can be used for `dur` to specify that we wish an element to continue playing indefinitely.

Instead of specifying a duration, we can specify a start and end time for the display of a media object, using attributes `begin` and `end`. Whereas durations are time differences, and therefore need no frame of reference, start and end points must be specified relative to some time origin. What this is depends on the enclosing synchronization element. For elements occurring within a `par`, times are measured relative to the start of the whole `par` element; for those occurring within a `seq`, they are measured relative to the end time of the preceding element. Hence, when we specified the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 452
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

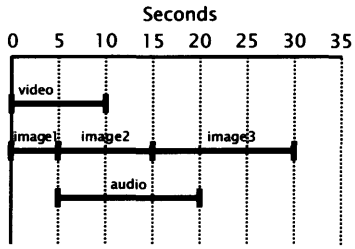


Figure 13.3
Effect of SMIL synchronization elements

begin attribute of the audio clip in the example above as 5s, this made the sound start playing five seconds after the beginning of the presentation (since the containing par element is the outermost synchronization element of the whole document); specifying its end as 20s truncates the clip's duration to fifteen seconds. Figure 13.3 illustrates the synchronization of this presentation on a timeline. The natural duration of the video clip, which is used in the absence of any explicit attributes, is 10 seconds.

⇨ The SMIL 1.0 specification includes rules to ensure that synchronization attributes can only be used sensibly. For example, an element's end value must not precede its begin.

The timing of the sequential elements in this presentation was specified purely by their duration, so they butt together in time. If we use begin and end, we can create pauses between them. For example, if we change the second img element as follows:

```
<img src="images/image2.jpeg"
     type="image/jpeg"
     region="upper"
     begin="5s" end="20s"/>
```

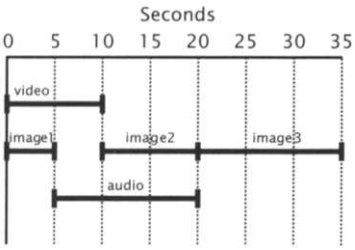


Figure 13.4
Introducing a delay in a seq element

there will be a five second gap between the display of image1 and image2 — because this is a seq element, the start time is not evaluated relative to the start of the synchronization element, but to the end of the preceding element. Figure 13.4 shows the effect of this change on the presentation's timeline.

⇨ The begin and end attributes can be thought of as specifying the position of SMIL elements on the presentation's timeline. Media elements such as video and audio clips or streamed text have their own internal timeline; the attributes clip-begin and clip-end can be used to specify offsets within an internal timeline, in order to extract a section from within a clip. For example, if the movie movies/fullmovie.mov was one minute long, the element

```
<video src="movies/fullmovie.mov"
       clip-begin="20s"
       clip-end="40s">
```

would define a twenty second clip from the middle of it. The use of these attributes removes the need to create a separate movie for the extract.

Elements within a par element can be synchronized directly with reference to each other. For this to be possible, each media element

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 453
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

must have an `id` attribute with a unique name as its value, which identifies the element in the same way as `id` attributes identify HTML document elements. A time value can then be specified in the form `id(`*element-id*`)(`*time-value*`)`, where the *time-value* can either be a clock value of the form just described, or one of the special forms `begin` or `end`. These two forms denote the start and end times of the element identified by *element-id*, respectively; the more general form denotes a time *time-value* later than the start of that element. For example, to delay the onset of the sound track of our example presentation until after the video had finished playing, we could make the following modifications:

```
<video src="movies/m1.mov"
       id="vid"
       type="video/quicktime"/>
<audio src="sounds/sound1"
       type="audio/aiff"
       begin="id(vid)(end)"/>
```

This way of specifying times, using what the SMIL specification calls an *event value*, is often more convenient than specifying them relative to the start of the enclosing synchronizing element; if the duration of some of the media elements is not known, it is essential. An event value can only be used by an element within the same synchronization element as the one it identifies. In the above example, since the `video` and `audio` elements are both in the content of the same `par`, the use of the event value `id(vid)(end)` as the beginning of the audio is legal. However, we could not use an event value `id(img2)(2s)` to specify that the audio should start two seconds after the display of the second image begins (assuming the second image element has `id img2`), because the image is inside the enclosed `seq` element. This restriction somewhat reduces the usefulness of event values.

When a `par` element contains elements that do not all finish at the same time, the question arises of when the whole `par` ends. Looking back at Figure 13.3, you will see that there are at least two reasonable candidates for the end time of the combination of video, images, and sound: at ten seconds — the earliest point at which one of the elements finishes, and at thirty seconds — the latest. Probably, the latter seems the more sensible interpretation, and that is what SMIL takes as the default: the end time of a `par` element is equal to the latest end time of any of its children (the elements it contains). The `endsync` attribute may be used to change

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 454
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

this behaviour. Its value may be `last`, which is the same as the default; `first`, which makes the end time of the `par` be equal to the earliest end time of any of its children; or an expression of the form `id(`*element-id*`)`, which makes the end of the `par` equal to the end of its child with `id` equal to *element-id*. For example, to make our presentation finish as soon as the video ended, we would change the start tag of the `par` element to:

```
<par endsync="first">
```

or

```
<par endsync="id(vid)">
```

If we use any `endsync` value other than `first`, what should happen to elements that terminate before the end of the `par`? Looking at Figure 13.3 again, you will see that there is a twenty second gap between the end of the video and the end of the whole presentation. What should happen to the video during this gap? Should it disappear from its allotted region, or should it be held frozen on its final frame? SMIL lets us choose between these two alternatives using the `fill` attribute, which can take either of the values `remove` or `freeze` (the default). To make our video disappear after it has reached its end, we would use the following tag:

```
<video src="movies/m1.mov"
       fill="remove"
       type="video/quicktime"/>
```

An alternative you might prefer to either of these options is to have the video start playing again from the beginning. Generally, you can arrange for any synchronization element to be repeated a specific number of times using the `repeat` attribute, whose value is a number, indicating the number of times the element is to play.[11] Since our video is ten seconds long, we can fill the entire presentation using three iterations of the video. To do this, we must put the `video` element inside its own `seq`, because repeat counts cannot be specified for media elements.

11
The semantics of English suggests that it should be one less, but in SMIL each iteration, including the first, is a repetition.

```
<seq repeat="3">
  <video src="movies/m1.mov"
         type="video/quicktime"/>
</seq>
```

The value of `repeat` can also be `indefinite`, which as you would expect, means that the element loops indefinitely — not, though,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 455
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

forever. If an element has an indefinite repeat count, it is ignored in the calculation of the duration of its enclosing synchronization element.[12] The effect of this is to ensure that it runs for as long as it needs to in order to 'fill up' an enclosing `par`. If, in our example, we had enclosed the `video` element in a `seq` with the following start tag:

```
<seq repeat="indefinite">
```

it would play until the last image had been displayed for its full duration, no matter how long the video clip was.

At first sight, although they may be awkward to work with, SMIL's synchronization elements and time-related attributes appear to be sufficient to allow a multimedia author to specify precisely any possible temporal relationship between the individual media elements of a presentation. It doesn't require much imagination to see how SMIL, or its equivalent, could be generated from a timeline of the sort we described earlier, so it would appear that implementing synchronized multimedia presentations is fairly trivial. In reality, things are not quite so rosy. SMIL's synchronization mechanism depends on the assumption that we can unambiguously measure the time taken to play any media element. For time-based media such as audio and video this is not the case. There are two distinct ways of measuring the time that elapses during playback. Consider a video clip that is intended to play at a rate of 15 frames per second. Assuming its `id` attribute's value is `clip1`, what time does the event value `id(clip1)(3s)` specify? Is it the time at which playback of the clip reaches its 46th frame, or is it three seconds after the clip starts to play, as measured by the clock of the computer system on which it is playing? There is no guarantee that the two will be the same, because there can be no guarantee that the clip will really play at the correct frame rate: the processor or disk may not be fast enough, or, if the clip is being streamed over a network, transmission delays may occur, causing some frames to be delivered too late. Hence, we must distinguish beween *media time*, the time measured by counting how much of the clip has been played, and *presentation time*, the elapsed time measured with reference to an external clock. SMIL clock values could be interpreted as either, producing different results. If they are interpreted as media time, then media elements wait for each other if a delay occurs; if they are interpreted as presentation time, synchronization may be lost, but the total running time of

12 Strictly, it is treated as if its end time were immediately after its start time.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 456
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

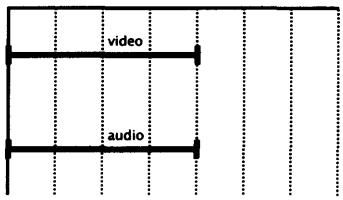


Figure 13.5
Video and audio perfectly synchronized

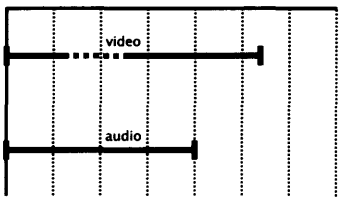


Figure 13.6
Soft synchronization

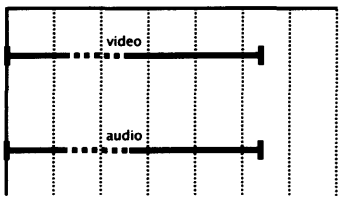


Figure 13.7
Hard sync: audio delay introduced

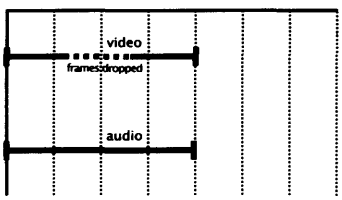


Figure 13.8
Hard sync: video frames dropped

a presentation will be maintained. The SMIL specification permits either interpretation, but recommends media time.

The same effect shows up in a different guise when we consider the moment-to-moment synchronization of elements within a `par`. If, for example, an audio clip and a video clip are playing simultaneously, we would expect them both to play at the same rate; in other words, media time in the two should pass at the same speed. If this is not the case, then the audio and video could slip relative to each other, leading to undesirable effects, such as loss of lip-sync, or the displacement of cuts in the video relative to the tempo of music on an accompanying sound track. On real computers and over real networks, independent delays can occur to either audio or video. Again, there is a choice between two possible responses, and a SMIL player is permitted to use either: it can use *hard synchronization*, with the elements within a `par` being synchronized to a common clock (thereby effectively matching their presentation times); or *soft synchronization*, with each having its own clock. Soft synchronization is really no synchronization, but it ensures that each element is played in its entirety. Hard synchronization maintains the temporal relationships between the elements (as nearly as is possible in the presence of delays), but distorts playback. Suppose, for example, that we have one video and one audio clip of identical durations, playing in parallel (see Figure 13.5), and that the video element experiences a delay of some sort during playback. If soft synchronization is used, the audio will carry on playing, and when the video resumes it will be out of sync (see Figure 13.6, where the dotted portion of the video line indicates a delay). To preserve hard synchronization, one of two things must be done. Either the audio must also be stopped until the delayed video data arrives (Figure 13.7), or some video frames must be dropped until the picture catches up with the sound again (Figure 13.8). The latter option is usually less intrusive, but, in the worst case, the picture may never catch up to the sound, so neither is very satisfactory.

⇨ The synchronization of separate tracks in a QuickTime movie, which we sketched in Chapter 12, is equivalent to hard synchronization, with frame dropping. Note that the use of the QuickTime time base is essential to detecting and correcting for loss of synchronization.

The problems just outlined are not a specific shortcoming of SMIL but are inevitable where synchronization is desired, but unpredictable delays can occur. Ideally, one would like to ensure

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 457
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

that delays cannot occur, or are kept so short that they do not interfere with playback. This can never be guaranteed, but we can assist by selecting versions of video and audio clips whose data rate is matched to the capabilities of the user's computer and network connection (the latter usually being the limiting factor). SMIL's `switch` element may be used for this purpose.

A `switch` may contain any number of other elements, each of which has one or more *test attributes*, which behave in a somewhat different way from other attributes. Their value is tested against some system parameter; if the test succeeds, the element may be selected and rendered. Only one of the elements within a `switch` is selected; they are considered in the order they appear in the document, and the first one whose test succeeds is chosen. The test attribute that is most relevant to our present discussion is `system-bitrate`, whose value is a number that is interpreted as a bandwidth in bits per second, and tested against the system bandwidth. This latter value may be simply taken from a user preference, for example specifying a modem connection rate, or it may be calculated from actual measurements of the system's performance. The mode of calculation is implementation-dependent. A test of this attribute succeeds if the available system bandwidth is greater than or equal to its value. Hence, a sequence of media elements can be provided in decreasing order of bandwidth requirements, with corresponding `system-bitrate` values, within a `switch`, so that a version can be chosen that is within the capabilities of the system. For example, we could prepare several different versions of a video clip, at different frame sizes and compression quality settings, and choose one of them as follows:

```
<switch>
    <video src="movies/mjepg-full-size-movie.mov"
           system-bitrate="56000"
           type="video/quicktime"/>
    <video src="movies/cinepak-full-size-movie.mov"
           system-bitrate="28800"
           type="video/quicktime"/>
    <video src="movies/cinepak-quarter-size-movie.mov"
           system-bitrate="14400"
           type="video/quicktime"/>
    <img src="images/movie-still.jpeg"
         system-bitrate="0"
         type="image/jpeg"/>
</switch>
```

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 458
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Note the final element in the `switch`: this is a default option (presumably the test will always succeed); a still image is selected, which does not place any strain on the capabilities of a low bandwidth system.

⇨ Other test attributes include `system-language`, which is used to select between different languages for text and speech, based on the user's preferences; `system-captions`, which can be set to the value on to display subtitling captions, for example, to supply a transcript of an audio sound track for the benefit of deaf people; `system-screen-size` and `system-screen-depth`, which are used to choose between different versions of images designed for particular screen sizes or colour depths.

We remarked earlier that the distinction between page-based and synchronization-based multimedia is not an absolute one: each style may possess characteristics of the other. As a prime example, SMIL provides simple uni-directional links, using linking elements modelled on HTML's anchors; however, the extra dimension of time that is central to SMIL calls for some refinement of the notion of linkage.

There are two linking element types in SMIL, the simpler being `a`, which behaves as a link source, much as HTML's `a` element type can. (Unlike HTML, SMIL does not use named `a` elements as link destinations, it makes use of `id` attributes instead.) The `href` attribute holds the URL of the link's destination, and clicking on the `a` element's content causes a jump to the destination, which will usually be another SMIL presentation. Formally, `a` elements have no synchronization properties and do not affect the temporal behaviour of any elements they contain. Since the content of an `a` is all that is actually displayed, and since the elements in the content will, in general, have a specified duration, the `a` element will behave as if it is only there while its content is being displayed. Consider the following simple presentation, for example.

```
<seq>
  <a href="presentation1.smil"
    <img src="images/image1.jpeg"
         type="image/jpeg"
         dur="15s"/>
  </a>
  <a href="presentation2.smil"
    <img src="images/image2.jpeg"
         type="image/jpeg"
```

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 459
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

```
        dur="15s"/>
  </a>
  <a href="presentation3.smil"
    <img src="images/image3.jpeg"
        type="image/jpeg"
        dur="15s"/>
  </a>
</seq>
```

Three images are displayed in sequence. Each of the images is contained in an `a` link, whose `href` points to a presentation — presumably one which expands on the content of the corresponding image. During the display of an image, clicking on it will activate the enclosing anchor, causing the linked presentation to begin playing. By default, the new presentation replaces the old one; this behaviour can be modified using the `show` attribute, which has one of the values `replace` (the default behaviour), `new`, which causes the new presentation to start up in a new window, leaving the old one playing in its original window, and `pause`, which causes the new presentation to start up in its own window, but leaves the old presentation paused at its current point, from which it resumes after the new presentation has finished. (Note how it is the temporal dimension of presentations that requires separate `replace` and `new` options.)

⇨ The SMIL 1.0 specification is actually rather vague about the duration of `a` elements, stating only that, " For synchronization purposes, the `a` element is transparent, i.e. it does not influence the synchronization of its child elements." The interpretation just given seems to be the only logical one, and corresponds to how the SMIL players available to us actually behave.

Like HTML, SMIL supports the use of fragment identifiers in URLs, so that links can point to individual elements (identified by the value of their `id` attribute) within a presentation. Again, the temporal dimension complicates matters. Consider a link such as the following:

```
<a href="presentation2.smil#vid1"
  <img src="images/image2.jpeg"
      type="image/jpeg"
      dur="15s"/>
</a>
```

and suppose that `presentation2.smil` has the following body:

```
<seq>
  <video src="movies/m1.mov"
         id="vid0"
         type="video/quicktime"/>
  <par>
    <seq>
      <img src="images/image1.jpeg"
           type="image/jpeg"
           id="img1"
           region="upper"
           dur="5s"/>
      <video src="movies/m1.mov"
             id="vid1"
             type="video/quicktime"/>
      <video src="movies/m1.mov"
             id="vid2"
             type="video/quicktime"/>
    </seq>
    <audio src="sounds/sound1"
           type="audio/aiff"
           begin="5s"/>
  </par>
</seq>
```

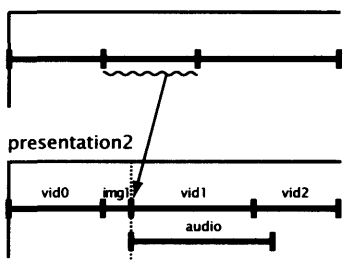


Figure 13.9
Effect of a fragment identifier

(The lower part of Figure 13.9 shows its timeline.) When the link is activated (at any time during the display of the image it contains, as shown in the diagram), a jump is made to the element with `id` equal to `vid1`. This element is not displayed until five seconds into the `par` element that follows the first video clip (`vid0`), so the effect of the jump must be to start `presentation2.smil` at that point, skipping the initial video clip and image. In general, when the destination URL of a link has a fragment identifier, the effect of following the link is to start the presentation containing the element it identifies, but as if the user had fast-forwarded to the point at which that element begins. Note that, where the element in question is within a `par`, all of the elements that are specified to play at the same time will be displayed. Hence, in the example just given, when the link is followed the sound will play as well as the video clip `vid1`.

⇨ If the destination of a link is an element with a `repeat` attribute, the presentation is started before the first iteration of the element. If it is contained within an element with a `repeat` attribute (which might itself be inside an element with a `repeat` attribute, and so on) matters become slightly complicated. The SMIL 1.0 specification describes how this case should be handled to ensure that the presentation behaves as expected.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 461
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

SMIL's a elements, in conjunction with fragment identifiers, provide links between entire elements; in other words, they connect identifiable points on the timelines of presentations. SMIL allows a more refined sort of link, based on identifying regions within the individual timelines of media elements. It provides the anchor element type for this purpose. These elements are the only ones that can appear in the content of a media element. (Previously we have always supposed that media elements were empty.)

An anchor is like an HTML A element, in that it may serve as either the source or the destination of a hyperlink, or both. The element type is furnished with href and id attributes for these purposes. It may also have a coords attribute, whose value is a set of four length values (with the same syntactical possibilities as in CSS), separated by commas. These define a rectangular area: the first two numbers are the upper left, the last two the lower right corner. An anchor with coords and an href thus defines a hot spot within the region in which the media element that contains it is displayed.[13] Several such anchors behave in the same way as an image map in HTML, although they can be associated with other media besides images.

An image map divides up the spatial region occupied by an object, associating links with different sub-regions. Since SMIL elements have a temporal dimension, it is also possible to divide up the time occupied by an object, and associate links with different intervals. The begin and end attributes of the anchor element type are used for this purpose. Suppose, for example, that the file trailers.mov contains a thirty second movie, consisting of three ten second segments, each of which is a trailer for some other movie (say an original Hollywood blockbuster and its two sequels). The following SMIL code will cause the trailer movie to play indefinitely; during each segment, if a user clicks on it, the corresponding full movie will be streamed from a video server.[14] Notice that, for the first time, the video element has separate start and end tags, because it is not empty.

13 The four values are subject to constraints that ensure that the rectangle's top is above its bottom, and so on. The rectangle is clipped to the media element's region.

14 See Chapter 15 for an explanation of rtsp://

```
<video src="trailers.mov"
       type="video/quicktime"
       repeat="indefinite">
  <anchor href="rtsp://movies.com/blockbuster.mov"
          id="trailer1"
          begin="0s" end="10s"/>
  <anchor href="rtsp://movies.com/first_sequel.mov"
          id="trailer2"
```

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 462
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

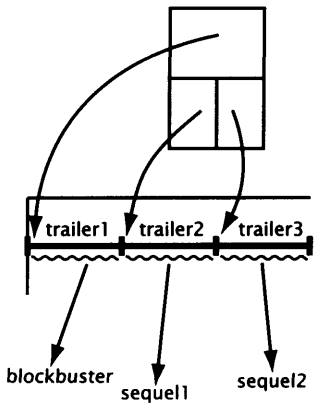


Figure 13.10
Anchors as link sources and destinations

```
          begin="10s" end="20s"/>
  <anchor href="rtsp://movies.com/second_sequel.mov"
          id="trailer3"
          begin="20s" end="30s"/>
</video>
```

The `id` attributes of `anchor` elements can be used as fragment identifiers in URLs, so that segments of a movie can be used as link destinations. For example, if the previous example appears in a file called `trailers.smil`, a presentation in the same directory could include an image map built out of anchors, such as the following:

```
<img src="poster.jpeg"
     type="image/jpeg">
  <anchor href="trailers.smil#trailer1"
          coords="0, 0, 100%, 50%"/>
  <anchor href="trailers.smil#trailer2"
          coords="0, 50%, 50%, 100%"/>
  <anchor href="trailers.smil#trailer3"
          coords="50%, 50%, 100%, 100%"/>
</img>
```

Assuming that the top half of the image holds a still from the original blockbuster, while the bottom half is divided into two, showing stills from the two sequels, clicking on a still will cause the trailers video to start playing at the beginning of the trailer for the appropriate movie (see Figure 13.10).

HTML+Time

SMIL provides a mechanism for producing synchronized presentations that can be played over the Internet, but it is largely separate from the World Wide Web. SMIL presentations can be embedded in Web pages and played with the aid of a suitable plug-in, much as video clips can, and HTML can be embedded in SMIL using `text` elements in a similar way, but the two languages cannot be truly integrated. For example, paragraphs within an HTML document cannot be arranged in time so that they are displayed in sequence.[15] *HTML+Time* (the 'Time' part stands for 'Timed Interactive Multimedia Extensions') is a proposal for applying the synchronization facilities of SMIL directly to HTML document elements. HTML+Time has not presently been endorsed by the W^{3}C, but it does have the influential backing of Microsoft and other major companies.

15
See Chapter 14 for a description of a mechanism whereby such effects can be achieved by another means, albeit with more effort.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 463
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

HTML+Time's basic idea is a simple one: additional attributes are introduced for every document element that can appear in the body of an HTML document. These attributes include `begin`, `end`, and `dur`, which take clock values in the same style as SMIL. Their effect on most document elements is to restrict their visibility to the specified time. When applied to those HTML elements that control style, such as `EM`, or to an element that uses the `style` attribute for in-line stylesheet information, the timing attributes control the duration of the indicated style. SMIL's event values cannot be used for `begin` and `end`. Instead, HTML+Time provides an attribute `beginWith`, which has as its value the `id` of some other element; clock values used for other attributes of the same element are interpreted relative to the start of the element with that id. For example,

```
<P id="start" begin="5s">
 content of start paragraph
 </P>
<P beginWith="start" begin="10s">
 text that appears 10 seconds
   after the start paragraph
 </P>
```

To produce the effect of a SMIL `par` element, an attribute `par` can be used with `DIV` or `SPAN`. Attributes corresponding to those that SMIL provides for its `par` element type can be used in conjunction with the `par` attribute to achieve similar effects. Inconsistently, HTML+Time does not offer a `seq` attribute, but instead adds SMIL's `seq` *element type* to HTML.

Since HTML+Time is only a proposal that has been submitted to the W^3C for comment, it would be unfair to judge it by the same criteria that are applied to mature W^3C Recommendations. The idea of adding synchronization facilities to HTML is potentially a productive one. Certain time-based effects, such as slide shows, which can only presently be achieved by stepping outside the declarative framework of HTML, may be produced relatively easily within the extended framework of HTML+Time. However, extending HTML goes against current trends in the World Wide Web. Stylistic properties have been separated from content, using stylesheets; tying temporal properties to content seems like a backward step. A 'temporal stylesheet' approach would seem to be more in keeping with the spirit of structural markup, and would have the additional advantage that the temporal specification mechanism would not

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 464
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

necessarily be tied to HTML alone, but could be applied to XML and other markup languages based on it.

⇨ The HTML+Time proposal addresses another of SMIL's shortcomings in passing only, without offering any improvement: the failure to provide any form of transitions, such as dissolves or page-turns, between elements of a presentation. These effects are standard fare in the mainstream, timeline-based, presentation tools. Without them, SMIL presentations have an abruptness that can be disconcerting, and usually makes them look distinctly amateurish.

Accessibility

Images, movies and sound may provide vital content in a multimedia production. Even if plain text can convey all the essential content, adding these other media can add interest, transmit the information more effectively, and make navigation more intuitive, but only if the images and video can be seen and the sounds can be heard. Not everybody can see and hear; not everybody who can see and hear can do so easily, or always understand what they see or hear. One of the great benefits that computers can bring to the world is in helping people who are blind, partially sighted, or deaf, or have cognitive difficulties or other physical disabilities. Special hardware and software can render text as speech or braille, or provide magnification; speech can be transcribed into text; information can be presented at a rate adapted to the user's abilities. All of this can be derailed by thoughtless use of multimedia elements. The W^{3}C has made strenuous efforts to ensure that, wherever possible, their recommendations include facilities for making information on the Web accessible to everybody with an Internet connection, irrespective of disability.

The use of images as links is a common example of an idiom that can frustrate users with poor eyesight if it used thoughtlessly. Unless some alternative is provided, any semantic content attached to the icons is lost to people who cannot see them. This does not mean, though, that all Web pages should revert to pure text in the interests of accessibility. HTML provides facilities that allows you to use multimedia elements, but still cater for people who, for some reason, are unable to perceive them. As the W^{3}C's accessibility guidelines for page authors put it:

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 465
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

> "Accessibility does not mean minimal page design, it means thoughtful page design. ...In general, authors should not be discouraged from using multimedia, but rather should use it in a manner which ensures that the material they publish is accessible to the widest possible audience."

⇨ Although questions of accessibility are particularly compelling in connection with disabilities, it is worth remembering that people might access Web sites using slow modems, so that the time taken to download images is not acceptable, or via hand-held mobile devices with screens too small to display images. Many PCs used in business do not sport sound cards. There are countless reasons why images and sounds might not be perceived, and it is only sensible to design your multimedia productions so that, as far as possible, they can be used — and make sense — without them.

Making your work accessible to the hard of hearing is not difficult, provided you make some effort: transcripts of speech and song lyrics can be provided;[16] alert sounds can be supplemented by visual cues, such as flashing icons; if you use speech to interact with users (as in some games involving dialogue) there should be an option to replace it with captions or speech bubbles.

16 QuickTime 3 and higher allows captions and video descriptions to be added to the clip itself.

The heavy use of graphically-based elements in multimedia means that almost every multimedia producer should be making some effort to compensate for visual problems. This means providing some alternative, usually in a textual form that can be rendered by a non-visual user agent, for every graphic element. This alternative must perform the same function as the graphic, which is not the same as simply describing it. For example, a navigational button requires a textual link, containing whatever clue the button's icon gave to the destination of the link, so a button labelled with the conventional icon of a house, denoting a link to a home page should not be augmented with some text saying 'little house on a red background', but a link labelled 'home'.

Not all authoring tools and formats provide much help, but HTML 4 has features especially designed to assist in making Web pages accessible to everybody. Chief among these is the `alt` attribute of `IMG` and `AREA` elements. Its value is a text string that serves as a substitute for the element if it cannot be rendered normally. So if the HTML is being rendered by a non-visual user agent, such as a speech synthesizer, or by a Web browser with images disabled, the `alt` value will be available in place of the image; in the case

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 466
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

of AREA elements, the `alt` value provides a textual link instead of the corresponding hot spot in the image map. The HTML 4.0 specification stipulates that `alt` must always be supplied with IMG and AREA elements; of course, Web browsers do little or no checking of the legality of HTML, but HTML verifiers should insist on the presence of these attributes.[17] Since `alt` text will be rendered in place, it should be relatively short. If a longer description of an image is required, the `longdesc` attribute can be set to its URL. In that case, the `alt` value should ideally provide enough information for a user to decide whether to pursue the link and read the long description. For example, the image map example of the previous section could have been improved like this:

17
This does not prevent graphics and Web authoring tools from routinely failing to generate them.

```
<IMG src="flower1.jpeg" usemap="#image-map-1"
          alt="Flower image map"
          longdesc="flowermap.html">
```

where the file `flowermap.html` provided a text-based set of links to the same destinations as those offered by the image map.

Several other attributes introduced in HTML 4.0 can be used to provide information that enables a user agent to help people whose disabilities make 'normal' browsing difficult. Consult the specification and the Web Accessibility Initiative's guidelines[18] for details. Unfortunately, Web browser vendors seem less interested in fully implementing these features than in adding marginally useful user interface gimmicks, so it is advisable to provide redundant textual links and other information to ensure that the graphical content of your Web pages does not make them inaccessible.

18
See the *Further Information* at the end of the chapter.

We alluded in Chapter 8 to one other way of enhancing accessibility: use structural markup to separate content from appearance. If a page is consistently tagged to reflect the semantics of its content, software can use the information in the tags to assist with presenting it in different media. More helpfully still, if appearance is controlled by a stylesheet, alternative stylesheet information can be provided for non-visual user agents. CSS2 proposes several properties to control aural presentation, such as volume and voice characteristics including stress, pitch and speech rate, and the use of 'aural icons' as cues. Consult the CSS proposal for more details.

The techniques we have just described can be complemented by use of the `title` attribute (not to be confused with the TITLE element type), which is an attribute of almost all elements in HTML 4.0. Its function is to provide a short description of the element. Its value is

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 467
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

not rendered as part of the page's content, but it may be displayed by a user agent in the style of a 'tool tip' or 'help balloon' when the cursor is over the element. Since titles can be spoken by non-visual user agents, this information can be made available to all users in some form. This can be helpful, since it enables a user to scan all the elements of a page and quickly get an idea of their function within the document. In particular, the title of a link may be used to indicate what the link is for.

SMIL shares some of the features in HTML that are intended to improve accessibility: all media elements support `alt` and `longdesc` attributes, linking elements have a `title`, while `par` and `seq` have an `abstract` attribute, which can be used to provide a textual description of their contents. However, a shortcoming of SMIL in this respect is that it offers no mechanism for providing users with control over the playback speed of presentations. Users with restricted movement or cognitive difficulties may not be able to respond quickly enough to anchors with limited duration, for example. An option to slow down a presentation would assist such users;[19] it is not clear how easy it would be to incorporate variable playback speeds into SMIL's model of time and synchronization.

19 Some SMIL players provide a means of pausing playback, but that is not what is required.

Further Information

The HTML references given at the end of Chapter 8 describe the use of HTML for hypermedia. [SMI] and [HTMb] are the definitions of SMIL and HTML+Time, respectively. [WAI] gives the Web Accessibility Initiative's guidelines for authors. [LH99] looks at the issues surrounding hypermedia development, which we have not considered. The use of scripting with combined media is the subject of the next chapter.

Exercises

The best way to understand combined media is to implement some multimedia productions. We suggest that you look at the chapter entitled *Multimedia Practice* for some suggestions. The following short exercises are merely concerned with some details of the languages described in the present chapter.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 468
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

1. An AREA element may have an attribute `nohref`, which, if true, specifies that the designated area has no URL associated with it. Since no part of the image is associated with a URL unless it is within the shape defined by some AREA, what possible use could you make of this attribute?

2. Explain how you would use SMIL's synchronization elements to make a simple looping animation from a sequence of still images. How would you add a synchronized sound track?

3. In the absence of any explicit `coords`, `begin` and `end` attributes, a SMIL `anchor` element is associated with the entire media element it is contained in. Does this make SMIL's `a` element type redundant?

4. What difficulties would somebody who was not able to use a mouse experience in trying to access the World Wide Web? What attributes introduced in HTML 4.0 can be used to help such people, and how?[20]

20 You will have to read the HTML 4.0 specification or a good HTML book.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 469
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Events, Scripts and Interactivity

14

The graphical user interface provided by contemporary operating systems is a paradigm of an *event-driven* system: the user does something (double-clicks an icon, for example), and something happens in response (an application program is launched). Hypermedia navigation, as we described it in Chapter 9, can be seen as a special case of this general model of interaction: clicking on some highlighted text, an image, or a hot spot in an image map, causes the destination of a link to be displayed. While this form of interaction can take you a long way, incorporating more general event-driven interaction into a multimedia production opens up additional possibilities.

The idea is to associate *actions* with *events*. We first need to identify a set of events. Most of these are initiated by the user — mouse clicks and movements, key presses, and so on — but some events are generated internally, for example, when a movie finishes playing, or simply after a specific period of time has elapsed.[1] The set of available events is not fixed, but depends on the particular combination of hardware and software being used. As well as identifying the set of events, we need to be able to specify actions. Some multimedia environments provide a pre-defined set of actions

1 There is often a *null* or *idle* event that is generated when nothing else happens.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 470
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

(often called *behaviours*) to perform common tasks, such as opening a file, playing a movie or replacing one image with another. More sophisticated environments let you define your own, by providing a *scripting language*: a small programming language with facilities for controlling user interface elements and multimedia objects. You write a script that performs the desired action when it is executed; you then associate that script with an event, using a tag or some command in an authoring environment, so that it gets executed when the user clicks on a particular image, for example.

➲ Here, we have assumed that actions can be associated with events, either by an authoring system, or using some tagging mechanism, and that they are executed under the control of some other program, such as a Web browser or a multimedia player. Such a program takes the form of a loop that is repeatedly executed. Every time round the loop, a check is made to see whether any event has occurred. The precise details of how this information is passed to the program depend on the operating system, but commonly, when an event occurs it causes an interrupt. The operating system then determines what event has occurred and assembles a data structure describing it — for example, if a mouse click occurs, this structure might contain the mouse coordinates and the state of the buttons and any relevant modifier keys — which it places in a queue before resuming the interrupted task. The event loop uses a system call to obtain the first event from the queue, and then takes a branch according to the type of event. Thus, associating an event with an action causes the branch corresponding to that event to execute the code you have written to perform the action. Sometimes, perhaps for efficiency reasons, it is not adequate to use the event loop and scripting language provided by a package, and you must write your own. The Java programming language provides for a flexible model of event handling that is used by the *Java Media Framework* to enable you to write programs that interact with multimedia objects.

One reason for associating actions with events is to provide interactivity: if the system can respond to events generated by the user, then the user can control the system's behaviour.[2] In particular, users can direct the flow of information that they receive. Events that occur at specific times provide another reason for using associated actions: they allow time-based behaviour and synchronization to be introduced into systems, such as the World Wide Web, that do not support it directly as part of their model of multimedia. Contrariwise, actions embedded in the timeline of a synchronized multimedia production can be used to provide non-

2 Within limits — see the discussion in Chapter 1.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 471
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

linearity. It is only a slight over-simplification to say that scripting can add the characteristic features of hypermedia to time-based presentations; it can add a temporal dimension to hypermedia; and it can add interactivity to both, so that multimedia productions that make use of scripts can provide the same experience to every user, no matter which model of media combination they start from.

Scripting Fundamentals

One definition of a *scripting language*, taken from the ECMAScript specification [ECM97], is

> "... a programming language that is used to manipulate, customize, and automate the facilities of an existing system."

The authors of the specification go on:

> "In such systems, useful functionality is already available through a user interface, and the scripting language is a mechanism for exposing that functionality to program control. In this way, the existing system is said to provide a *host environment* of objects and facilities which completes the capabilities of the scripting language."[3]

3 Our italics.

Scripting languages can be distinguished from mainstream programming languages such as C++ or Java, which provide some control structures, abstraction mechanisms, and built-in data types, such as numbers and pointers or references; the abstraction mechanisms of a modern programming language allow you to define your own data types, constructed out of the basic types provided by the language. A scripting language provides some control structures and a few basic types too — and may provide some abstraction mechanisms — but it also provides objects and data types that belong to some 'existing system'. For example, a scripting language for use with a relational database system will provide data types corresponding to relations and tuples; one for use as an operating system command language will provide data types corresponding to processes and files. (Note that here we are using the expression

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 472
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

'data types' in the modern sense of *abstract data types*: a set of values together with a set of operations.)

Our concern is with multimedia, so the scripting languages we are interested in must provide objects corresponding to the elements of a multimedia production. It is not adequate to provide a type for each medium — text, sound, video, and so on — the types of objects available need to take account of the way in which media objects are combined within the production. (Recall from the passage just cited that a scripting language provides program control equivalent to the user interface.) Thus, in a scripting language for use with XML or HTML, objects must be provided that correspond to document elements and to elements of the user interface, such as windows. In a language whose host environment is a time-line based system such as Flash, we need objects corresponding to frames and the various elements that may appear inside them.[4] The scripting language will allow us to perform computations on the attributes of these objects, or create new ones, thereby affecting their appearance and behaviour. This computation will be triggered by events. Some of these events will be initiated by the user, so scripted actions provide a means of allowing user input to affect the flow of control, or putting it another way, they allow certain elements of a multimedia production to function as controls providing interactivity . For example, an action may be attached to an image, with the effect that clicking on the image causes a particular movie to start playing in another window.

4
Although we talk of the host system providing a set of 'objects', a scripting language does not need to be explicitly object-oriented, although most are.

World Wide Web Client-Side Scripting

The World Wide Web offers two potential host environments for a scripting language — servers and browsers (clients) — giving rise to two types of Web scripting: *server-side* and *client-side*. Server-side scripting is used to enable an HTTP server to communicate with other resources, such as databases, and incorporate data obtained from them into its responses. In particular, server side scripts are used to enable a server to construct Web pages dynamically from time-varying data. We will discuss server-side scripting in Chapter 15. In this chapter, we will confine ourselves to client-side scripting, which is used to control the display of media elements that have been retrieved from a server. World Wide Web scripting on the client-side is a comparatively recent development, which has helped transform the appearance and behaviour of Web pages.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 473
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Executing a client-side script means allowing executable code that has been downloaded from the Internet to run on your machine. Since you have no real guarantee that the person whose code you have downloaded is not malicious, deranged, or in the pay of a market research organization, this would be a foolhardy thing to do if scripts were able to perform arbitrary computation. So they aren't. In particular, scripts running in a Web browser cannot access any local resources, such as files on your hard disks, or make any network connections, and their interaction with the server from which they originated is limited to requesting new resources and posting information from HTML forms.[5] As a result of these restrictions, client-side scripts are (reasonably) secure, but limited in the useful work they can do. In effect, they cannot do more than modify the browser's display in response to events. Generally, client-side scripts are used to provide feedback to users (for example, by changing the colour of a clickable item when the cursor is over it) or, more often, just to add interest to a site by providing a richer user interface than the simple one consisting of loading a new page when a link is clicked.[6]

5 To prevent abuses of this facility, there are additional restrictions on the elements of the browser state which a script can read.

6 Where forms are embedded in Web pages, client-side scripts are often used to validate input to fields, without the need to contact the server and run a CGI script.

The first scripting language for the World Wide Web was a proprietary product called LiveScript, which was embedded in Netscape's Navigator Web browser; its name was changed to JavaScript, in what is generally understood to be a marketing-inspired move to capitalize on the interest generated by Java — a largely unrelated programming language. Microsoft rapidly produced an almost compatible scripting language, disingenuously called JScript, for their Internet Explorer browser. In an attempt to prevent a Web scripting schism, the European Computer Manufacturers' Association (ECMA) was asked to produce a scripting language standard, based on JavaScript and JScript. ECMA did something smarter: they defined a 'core scripting language', called ECMAScript, which could be adapted to a variety of host environments — a Web browser being just one of them.

In order for ECMAScript to to be able to interact with a Web browser, there must be a well-defined interface between the two. Since ECMAScript is an object-based language, this can be provided in the form of a set of objects representing the elements of an HTML (or, in time, XML) document and any styles applied to them, and the components of the browser interface. The W^{3}C's *Document Object Model (DOM)* is intended to provide this interface. Unfortunately, at the time of writing, only part of the DOM has been defined, and the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 474
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

two major browsers use document models of their own, which are not entirely compatible. Nevertheless, it is possible to identify most of what an object model must look like, so that we can describe client-side Web scripting. You should be aware, though, that in practice a lot of effort and ingenuity is required to ensure that scripts will work as intended on every browser that supports client-side scripting.

⇨ Both Netscape and Microsoft claim to have brought their scripting languages into line with ECMAScript. In practice, therefore, 'JavaScript' ought to mean ECMAScript with Netscape's version of the DOM, and 'JScript' ECMAScript with Microsoft's, but 'JavaScript' is often used loosely to mean either. To complete the confusion, the expression *dynamic HTML*, which seems to have been devised by Microsoft to refer to the use of their object model and JScript or VBasic for client-side Web scripting, has been adopted by Netscape to refer to client-side scripting using the aspects of *their* object model that are concerned with style sheets and absolute positioning,[7] but is often used loosely to refer to anything to do with client-side scripting, style sheets, or both.

Although ECMAScript is a simple language, compared with C++ for example, it would still require more space than we have available in this book to provide a complete description of all its features and the way in which it can be used. Instead of attempting to do that, we will use a set of examples to try to provide an impression of what can be achieved and how, using client-side scripting to automate multimedia elements of Web pages.[8] In the interests of simplicity we will only introduce the minimum of syntax, so much of our coding will be less elegant than it could be — ECMAScript allows some sophisticated programming techniques to be employed by those who know how. Since the W^{3}C DOM remains incomplete and has not been implemented in any browser at the time of writing, we will present scripts using the set of host objects that works with existing fourth generation Web browsers, with a preference for Internet Explorer 4 where there are differences, since its object model is closer to the emerging DOM. The main difference between the current model and the Level 1 W^{3}C DOM is that, since the latter is designed to support XML as well as HTML, its object names are prefaced with `HTML`. Thus, where we refer to `document`, the standard, when complete, will refer to `HTMLdocument`, and so on. We expect that the old names will continue to work for some time, in the interests of backwards compatibility.

7 Which Netscape have helpfully implemented using their own <LAYER> tag instead of (latterly, as well as) the appropriate CSS2 properties.

8 Much of this will be concerned with manipulating image elements, so you should remember that whenever such techniques are employed, it is necessary to provide alternatives for non-visual user agents.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 475
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

In the following sections we assume some familiarity with programming and programming languages. If you are not comfortable with these ideas, you may still benefit from skimming through the rest of this chapter, just to get a feeling for what can be achieved with scripts.

ECMAScript Syntax Outline

If there is any justification for the similarity between the names JavaScript and Java, it is that the sytax of statements and expressions of JavaScript — now standardized in ECMAScript — closely resembles that of Java, and hence of C and C++. If you know any of those languages, the form of ECMAScript programs will contain few surprises. If you are only familiar with Visual Basic, or with the Algol and Pascal family, you may find the detailed concrete syntax slightly odd, but the statement and expression forms available will be roughly as you expect: assignment statements, a variety of loops and conditionals, function definition, call and return, and so on. A major difference between ECMAScript and Java, C++, Pascal and most other mainstream programming languages is that ECMAScript is an *untyped* language. To be precise (since the term 'untyped' does not have a unique meaning), a variable is not restricted, either by declaration or any other mechanism or convention, to holding values of just one type. The same variable can hold a string at one time, a number at another, and a Boolean at another.[9] There can therefore be no type checking. As is the case with other untyped languages, there is no need to declare variables before you use them, although it is good practice to do so.

9 Not that we recommend the practice.

The primitive values supplied by ECMAScript belong to the types `Boolean`, containing the two values `true` and `false`; `Number`, containing floating point numbers;[10] and `String`, containing strings of Unicode characters (although implementations commonly only support ISO 8859-1). The usual operations on these objects — Boolean algebra for `Boolean`, arithmetic for `Number`, and concatenation (denoted + in ECMAScript) for `String` — are provided. Primitive values can be combined into arrays and objects. This gives you the capability of creating your own data types, but doing so is only really appropriate if you are designing re-usable script libraries, perhaps to serve as behaviours (see page 466). Most of the time, casual scripts of the sort we will display in this section make use of ECMAScript's support for objects by manipulating its *built-in objects*

10 Double-precision 64-bit format IEEE 754 values.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 476
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

— the equivalent of a run-time library, including the `Math` and `Date` objects — and *host objects*, which provide the interface to the host environment — in the case we are considering, a Web browser.

Objects

ECMAScript is usually described as an *object-based* language. This means that it supports the creation and manipulation of objects, but these are not systematically organized into hierarchical classes as they are in an *object-oriented* language, such as Java. An object in ECMAScript is just a collection of named data items, known as *properties*,[11] and functions, known as *methods*. Any collection will do, and properties and methods can be added to or removed from an object at any time. If a variable called, for example, `the_object`, holds an object, you can access any property, such as `a_property` using the notation `the_object.a_property`, and call a method, `a_method`, with `the_object.a_method()`. If the method takes any arguments, they are written inside the brackets, separated by commas, as in `the_object.b_method('string', true)`.

11 Strictly speaking, the property is a container for the data item. This allows attributes to be associated with a property — such as whether it is read-only — in a semantically clean manner. Most of the time, the distinction is academic.



Objects are implemented simply as associative arrays, indexed by the method and property names. This works because, in ECMAScript, functions are 'first-class' values, and can be stored in an array like any other value. The apparently anarchic properties of ECMAScript objects follow from this implementation. If you are writing complex scripts that require you to define your own objects and you wish to impose some order on them, in the way you would do with classes in C++ or Java, you can do so by using a constructor to build them. Whenever a particular constructor *K* is called to build an object, it will always do the same thing, so, apart from any differences deriving from the arguments to *K*, the objects it initializes will be the same — they will have the same methods and properties, although the initial values of the properties may be different. We could very well say the objects built by *K* are all instances of the same class.

ECMAScript provides syntax for calling constructors when objects are created, and even supports a form of inheritance (based on object *prototypes*). Since we are not going to require these more advanced features in our example scripts we will not pursue these matters further. If you are interested in these features, or need them, consult the detailed references provided at the end of the chapter.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 477
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Among the host objects provided by Web browsers to allow scripts to manipulate HTML documents,[12] the `document` object plays a primary rôle. As you would expect from its name, it provides an interface to an HTML document. It contains properties holding the document's title and various pieces of information derived from the HTTP request used to access it, such as the URL and referrer; there are also properties which are arrays, containing all the links, images, and embedded objects in the document, and a very useful property called `all`, which is an object having, for each document element with an `id` attribute, a property with a matching name holding an object corresponding to the element.[13] For example, if a document contains a paragraph element with `id` value `'opening'`, the object `document.all.opening` will be an object describing that paragraph. The `document` object also has several methods, including `write`, which writes its argument into the current document.

12 We will not consider XML.

13 The W3C DOM specifies methods `getElementById` and `getElementByName` for accessing these objects, instead of making them into properties of the `document.all` object.

➲ Note that the `document` object is not the root of a parse tree of the document structure, as you might expect. The Microsoft object model does provide methods `children` and `contains` which can be used to navigate the document hierarchy, but these have not been incorporated into the W3C DOM.

Which document is the `document` object related to? It is the document currently being displayed when the script is executed. But how is a script to be executed in conjunction with a document? The HTML `SCRIPT` element is used to embed scripts in documents. When a page is loaded, all the scripts embedded in it are executed, with the `document` object providing its interface to that page. A simple example will make this clear. Consider the following:

```
<HTML>
<HEAD>
<TITLE>Dynamically Generated Content</TITLE>
</HEAD>

<BODY>
<SCRIPT type="text/javascript">
<!-- /* Hide content from old browsers */
document.write('<H1>', document.title, '</H1>');
// end hiding content from old browsers  -->
</SCRIPT>
</BODY>
</HTML>
```



Figure 14.1
Dynamically generated content

When this page is loaded into a browser, it produces the display shown in Figure 14.1. The interesting part is the SCRIPT element, enclosed between start and end tags in the usual way. The start tag sets the value of the type attribute to the MIME type `text/javascript` — other scripting languages may be used, so we must identify the one being used; despite the anachronism the `javascript` sub-type is the appropriate one. Next comes a trick, similar to that demonstrated in Chapter 8 to protect stylesheets from old browsers. There are two forms of ECMAScript comments: bracketed comments, delimited by the sequences `/*` and `*/`, and end of line comments, extending from the sequence `//` to the end of line. Both of these are used here in conjunction with HTML comments to hide the script from browsers that don't understand it. In this case, the script is just the single method call:

```
document.write('<H1>', document.title, '</H1>');
```

The script is executed at the point it is encountered by the Web browser as it parses the HTML document. The `write` method of the `document` object may take an arbitrary number of arguments. Here, we have given it three: the first and last are literal strings containing the start and end tags of an `H1` element; the middle argument is the value of the `title` property of the document, and this is a string containing the content of the TITLE element of this page, i.e. `'Dynamically Generated Content'`. The `write` method does what you probably expect, writes its argument into the document. Since the script is executed when it is encountered during the parsing of the document, the text it writes replaces the SCRIPT element, so it is as if the document's body were:

```
<H1>Dynamically Generated Content</H1>
```

and, indeed, the display produced looks exactly as if it was.

It doesn't take a very astute reader to recognize this script as totally pointless — we knew the text of the title and could have more economically embedded it directly in the document, in the form just shown, without executing any script. Now that we know how to combine script and document, via the SCRIPT element and the `document` object, though, we can go on to produce genuinely dynamic pages.

A clumsy but effective means of generating dynamic page content is to build it out of text solicited from the user. The function `prompt`[14] can be used for this purpose. It takes a single string argument,

14
Technically, `prompt` is a method belonging to the *window* object that is attached to the window in which the current document is displayed. All unqualified names (including *document*) are implicitly methods or properties of this object, so, nearly all the time, you do not need to be aware of it.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 479
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

and when called it causes a dialogue box to be displayed, with its argument as a prompt and a text entry field. Any string typed into this field is returned as the result of `prompt` when the user dismisses the dialogue. For example, the following code will cause the dialogue shown in Figure 14.2 to be displayed; if the OK button is clicked with the text entered as shown, the variable `their_name` will be set to the string `'Wilfred'`.

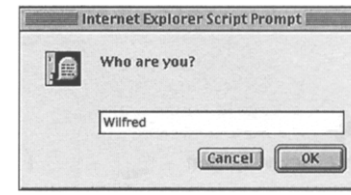


Figure 14.2
Prompting for user input

```
var their_name = prompt('Who are you?');
```

The value of this variable is in scope in all other script fragments embedded in the document after this call, so it can be used to put together some text incorporating the user's name, which can be put into the document with `document.write`. A good place for the call is in the document's HEAD. The script is executed — so the dialogue is displayed — when it is encountered. The HEAD is completely processed before the content of the BODY is displayed, so the prompt will appear before the page itself. If `prompt` is called from within the page BODY, the dialogue may appear in the middle of the page being rendered, which will usually be distracting. Hence, we might use a prompt in the following manner:[15]

15
We have suppressed the script-hiding comments, and will continue to do so from now on.

```
<HTML>
<HEAD>
<TITLE>Prompting</TITLE>
<SCRIPT type="text/javascript">
var their_name = prompt('Who are you?');
</SCRIPT>
</HEAD>

<BODY>
<H1>Interactively Generated Content</H1>
<HR>
<SCRIPT type="text/javascript">
document.write('<H2>Why a Duck?</H2>');
document.write('<P>Especially one called ',
                  their_name, '</P>');
</SCRIPT>
</BODY>
</HTML>
```

The dialogue box will appear as we saw previously, and then the page shown in Figure 14.3 will be displayed.

⇨ You probably want to know what happens if the user clicks the `cancel` button in the dialogue box. An `undefined` object is

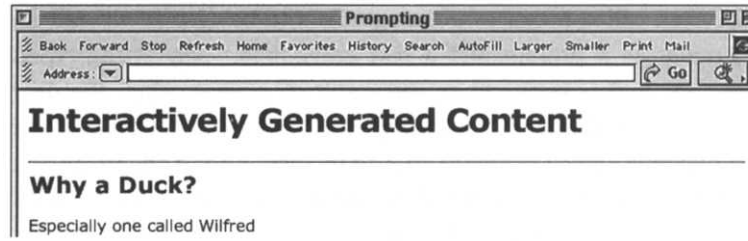


Figure 14.3
Interactively generated content

returned, and production code should check for this eventuality and do something sensible. If you want to see what happens without any check, try it.

This example is fatuous, but it demonstrates a serious point: information obtained on the client machine at the time the data is downloaded from the server is incorporated dynamically into the page when it is displayed. In other words, interaction with the user has influenced the appearance of the page.

Event Handlers

Much of the interaction required from multimedia presentations and Web pages is concerned with controlling the flow of information. This can be achieved gracefully by using on-screen controls — icons, images, or text — that respond to user events, such as mouse clicks. A simple mechanism for associating a script with an event and a document element is provided by HTML 4. Almost any document element can have attributes whose name identifies a class of events and whose value is a short piece of code to be executed when the event occurs to that element. Table 14.1 shows the names of the event-related attributes[16] that can be used with most elements, and the events that cause the script that is their value to be executed. Certain elements (for example HEAD) cannot sensibly have these attributes; others, particularly those representing HTML forms may have extra attributes, but we will not go into the detail of these here, since forms are not often used in conjunction with multimedia pages.[17]

The code which is the value of an event-related attribute is known as an *event handler*. It is normal for it to be a call to a function. This way, the code that must be included in line with the element tag is compact, and that which performs the actions can be kept separate

16
The eccentric use of capital letters is a common convention for showing the component words of these names. It has no actual significance — HTML is insensitive to the case in which attribute names are written.

17
Note, though, that in versions of HTML earlier than 4.0, *only* form elements had any event handlers, so they were often used for purposes other than data entry where event handling was required.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 481
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

and easy to read. In this case, we usually refer to the function as the event handler.

One use of event handlers has become a cliché of client-side Web scripting: the 'rollover button'. This is a button — i.e. a small image that serves as a control — whose appearance changes when the cursor is moved over it, to indicate that a mouse click at that point will activate it. This provides some useful feedback to the user, if it is not over-done. The essence of the effect is captured in the following example.

The change of appearance is achieved simply by assigning to the `src` property of an object corresponding to an image. As we mentioned earlier, the `document.all` object has a property for each element with an `id` attribute. The following two event handlers update the `src` of an element with `id` equal to `circlesquare`, the first setting it to a GIF whose name suggests it is a picture of a circle, the second to a similar image of a square.

```
function in_image() {
    document.all.circlesquare.src = 'images/circle.gif';
}

function out_of_image() {
    document.all.circlesquare.src = 'images/square.gif';
}
```

These definitions would be placed in the `HEAD` of the document. The names of the functions have no semantic significance; they are chosen purely to indicate their function to human readers. They are associated with events in the `IMG` tag of the `circlesquare` element:

```
<IMG src="images/square.gif" alt="square or circle"
  onMouseOver="in_image()" onMouseOut="out_of_image()"
  id="circlesquare">
```

Setting the value of the `onMouseOver` attribute to the string `"in_image()"` ensures that whenever the cursor is moved over the image, the function `in_image` will be called, to set the image to a circle. Similarly, the `onMouseOut` attribute has been set to call `out_image` to make the image revert to the original square. The combination of these handlers ensures that the image displays as a circle whenever the cursor is within its bounds, and as a square the rest of the time. In a practical situation, the images would portray, for example, a push-button in different colours, to indicate readiness when the cursor was over it. Additional `onMouseDown`

Table 14.1
HTML 4.0 Event Handlers

`onClick`	a mouse button was clicked
`onDblClick`	a mouse button was double-clicked
`onKeyDown`	a key was pressed down
`onKeyPress`	a key was pressed and released
`onKeyUp`	a key was released
`onMouseDown`	a mouse button was pressed down
`onMouseMove`	the cursor was moved
`onMouseOut`	the cursor was moved away from the element
`onMouseOver`	the cursor was moved over the element
`onMouseUp`	a mouse button was released

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 482
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

and onMouseUp handlers could be defined to show the button being pressed and released.

The technique of updating the src property of an image object can be used to add animation to a Web page, under the control of a script. This is an improvement on the use of animated GIFs, since these cannot be controlled: they play once on loading, or for a specific number of times or forever in a loop. Using a script, we can arrange for an animation to be started and stopped by a control or some other means. In the following example, we make a simple animation play only as long as the cursor is over it — a moving version of the rollover we just described.

The strategy is easily grasped, but a number of tricks are required to make it work properly. We want to associate a sequence of frames — we will use ten — with an image element, and cycle through them in turn. The onMouseOver event handler for our image will start the cycle going, and the onMouseOut will stop it. We can write the image tag straight away:

```
<IMG src="images/animation/frame1.gif"
     alt="simple animation"
     id="the_image"
     onMouseOver="start_animation()"
     onMouseOut="stop_animation()">
```

We are assuming that the images for the animated sequence are in a directory images/animation below the document, as files named frame1.gif, frame2.gif, ... frame10.gif. The first frame is displayed in the image element with id the_image when the page is loaded; the two relevant handlers are set to call functions to start and stop the animation. All we have to do is define those functions, and supply any supporting code they require. The appropriate place to do so is in the document's HEAD.

We know that what we need to do is assign a new value to the_image.src at regular intervals — let us say every eighty milliseconds, to give a frame rate of 12.5 fps — and we know what those values should be (the URLs of the frames of the animation). If we assume that we have a function animate that does that as long as a Boolean variable continuing is true, we can define the handlers in terms of it:

```
function start_animation()
{
   continuing = true;
```

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 483
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

```
    animate();
}

function stop_animation()
{
    continuing = false;
}
```

We don't need to declare `continuing` — undeclared or uninitialized variables have an undefined value, which evaluates to false in a Boolean context, when they are first used — but we can if we like (before the functions):

```
var continuing = false;
```

Our problems now seem to be concentrated in the `animate` function. We know from the rollover example how to update images, and it's fairly clear how to use a counter variable to cycle through the frames, but how are we to produce the necessary delay between them? The answer lies in a built-in function that we have not yet described: `setTimeout`. This takes two arguments. The first is a string of code, the second is a number, which will be treated as a time interval in milliseconds. The effect of `setTimeout(`*code*`,` *delay*`)`, is to execute *code* after a delay of *delay* milliseconds. In order to repeat an action at regular intervals, all we need to do is make *code* be a call to a function that calls `setTimeout` with a call to itself as the first argument. This is what our `animate` function does. We need some globals first:

```
var delay = 80;
var num_frames = 10;
var i = 0;
```

Notice that two of these are really just constants, but ECMAScript does not provide anything like C++'s `const`, so if you want named constants you must use variables. Now we can write `animate`:

```
function animate() {
    document.all.the_image.src =  'images/animation/frame'
                                    + (++i) + '.gif';
    if (i == num_frames)
        i = 0;
   if (continuing)
        setTimeout('animate()', delay);
}
```

The expression on the right of the first assignment builds the name of the next frame using the value of the variable `i` as the frame number — the prefix ++ operator pre-increments the value, which ensures that it cycles correctly through the frames. The test for equality with `num_frames` is used to make the value wrap round to zero when it reaches the last frame number, so that we do truly cycle. The last conditional is the key: we test `continuing` to see whether to go on — remember that this variable is set to false when `stop_animation` is called. If we are continuing, we set a timeout, so that `animate` is called again after another delay.



The code just given works after a fashion, but has a major defect: the first time a URL is assigned to the image's `src`, the image file has to be fetched over the network, which may cause an appreciable delay, so the first cycle of animation may be slower than intended. Subsequently, the file will be cached, and loaded much faster. We would like to be able to preload the images so that every cycle runs at the proper speed. The way this is usually done is by constructing an array of `Image` objects, and populating it with the frames of the animation:

```
frames = new Array;
for (var j = 0; j < num_frames; ++j)
{
    frames[j] = new Image;
    frames[j].src = 'images/animation/frame'
                            + (j + 1) + '.gif';
}
```

(Array indices start at zero.) Although these images are not displayed, this code is sufficent to cause them to be downloaded and cached. We can now copy the `src` of one of the images in the array when performing the animation:

```
document.all.the_image.src = frames[i++].src;
```

which is slightly faster than assigning the URL directly and means that we only have to change the initialization code to get a different animation.

Scripts and Stylesheets

In a well-designed Web page, content is separated from appearance, with the latter being controlled by stylesheets. In order to be able to make changes to the appearance, scripts need to be able to

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 485
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

manipulate stylesheets. This is possible, but here, above all, there is no standard model that can be used: Level 2 of the W3C DOM, which deals with styles, is only a first draft as we write, and there are significant differences between the available implementations. Conceptually, though, there is no great difficulty, so although the details of the relevant objects may change, the general approach to controlling appearance with scripts should remain roughly as we describe it.[18]

18
Our description is of the DOM as implemented by Internet Explorer 4.

There are two ways in which we can change styles: either by changing the style applied to a particular document element, or by altering the stylesheet applied to an entire document. The former is the easier option: each object in `document.all` has a `style` property, which is itself an object with properties (in the ECMAScript sense) corresponding to the CSS properties applied to the element. By assigning to these, its appearance can be changed. Suppose, for example, a document has the following stylesheet applied to it:

```
H1 {
      color: lime;
}
H2 {
      color: blue;
}
```

Then the following HTML element will produce a lime-coloured header:

```
<H1 id="intro">Introduction to Limes</H1>
```

Its colour can be reset to a more sober value by the following assignment, which might be put into an event handler:

```
document.all.intro.style.color = 'black';
```

Altering the stylesheet itself is slightly more complicated. The `document` object has a property `styleSheets`, which is an array containing an object for each STYLE element in the document. This in turn includes an array `rules` containing an object for each rule in the STYLE element. Each rule object has a `selectorText` property, which allows you to extract the selectors from the corresponding rule, and a `style` property, which provides access to the values of the CSS properties set by the rule. By assigning to these properties,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 486
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

changes to the styles affecting the entire document can be made. For example, executing

```
document.styleSheets[0].rules[1].style.color = "fuchsia";
```

would change the colour of every level 2 header in the document. Access to STYLE elements and the rules within them is by numerical index, *starting at zero.* Here, `styleSheets[0]` is the first STYLE; assuming its content is the rules shown above, then `styleSheets[0].rules[1]` is the object corresponding to

```
H2 {
      color: blue;
}
```

so our assignment has the same effect as transforming this rule into

```
H2 {
      color: fuchsia;
}
```

and applying it, with garish results.

Combining the scriptability of styles with absolute positioning enables you to move document elements about the screen. In conjunction with the use of `setTimeout` described earlier, this allows you, among other things, to animate text, in a style often favoured for the opening credits of television news bulletins and sports programmes for example.

⇨ Because absolutely positioned elements with different `z-order` values behave rather like layers in Photoshop, and because the first implementation of absolute positioning that was available was based on Netscape's proprietary (and now deprecated) `<LAYER>` tag, it has become customary to refer to the process of moving absolutely positioned elements with a script as 'animating layers'.

As a crude example of text animation, we will construct a Web page in which the word MOTION moves diagonally from top left to bottom right; when it has gone a certain distance, it stops, and the word BLUR appears superimposed over it for a while, before a new page is loaded.[19] The body of the document looks like this:

19
If you try this, you'll see why we chose the text 'MOTION BLUR'.

```
<SPAN ID="layer1">MOTION</SPAN>
<SPAN ID="layer2">BLUR</SPAN>
```

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 487
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

```
<SCRIPT TYPE="text/javascript">
   move_the_text();
</SCRIPT>
```

Each word is enclosed in a SPAN element with its own `id`, so that we can apply positioning style to it. The script will be executed when the page is loaded; it calls a function to do the work. Before looking at that function, we need to see the stylesheet applied to the two spans.

```
<STYLE TYPE="text/css">
#layer1 {
    position: absolute;
    top: 35;
    left: 40;
    font: x-large ;
    color: blue;
}

#layer2 {
    position: absolute;
    top: 300;
    left: 540;
    font: x-large ;
    color: red;
    visibility: hidden;
}
</STYLE>
```

Recall that we use fragment identifiers as selectors to apply a style to a specific element. The first rule thus positions the first layer (MOTION) at $(40, 35)$, using absolute positioning; an extra large blue font is used to make the word conspicuous. The same font, in red, is used for the other layer, which we position down and to the right at $(540, 300)$ (vertical coordinates increase downwards); the `visibility` property is set to `hidden` — we do not want to see BLUR until MOTION has stopped moving. We do not bother to specify `z-order` explicitly; the elements are stacked by default in the order they appear in the stylesheet.

Now to make things move. We use exactly the same technique as we used to animate a sequence of images: the function `move_the_text` updates the `top` and `left` properties of the `layer1` object's `style`, thereby moving it to a new position, and then sets a timeout, with itself as the code to be executed after the delay. To ensure that the process terminates, we use a counter `num_steps`, initialized to

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 488
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

50 and decremented until it reaches zero. At that point, we make layer2 visible. A feature we have not seen before is then used: the `location` object[20] has a property called `href`, which holds the URL of the document currently being displayed. Updating this property causes a new page to be loaded. We want a pause — so the user can admire the final state of the animation — before this happens, so we set a timeout, this time to a longer value. Here is the complete code of the function, with its associated variables:

20 Actually a property of `window`.

```
var delay = 40;
var num_steps = 50;
var x = 35, y = 40;
function move_the_text() {
    x += 10; y += 5;
    document.all.layer1.style.left = x;
    document.all.layer1.style.top = y;
    if (--num_steps)
        setTimeout('move_the_text()', delay);
    else
    {
       layer2.style.visibility = 'visible';
       setTimeout('location.href = "image-swap.html"',
                                              delay*35);
    }
}
```

We should emphasize again that in these examples we have only sought to demonstrate how a scripting language in conjunction with a document object model can be used to add interactivity and dynamic effects to Web pages. We have not tried to demonstrate good programming style, nor to address the real practical issues concerning compatibility between different browsers and different versions of browsers.

Behaviours

Scripts offer the potential for controlling the user interface to a multimedia production on the basis of elaborate computation, carried out using most of the facilities of a modern programming language. Much of the time, authors only require a limited repertoire of interface enhancements and animation facilities, such as those illustrated in the previous section.[21] If an authoring system

21 Since user interfaces are subject to fashion, the desired repertoire may change over time.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 489
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

can provide a set of parameterized actions that is adequate to satisfy most needs, and a suitable interface for attaching them to elements of a multimedia production, then much of the necessity for scripting is removed. This means that designers can add interactivity and animation to their productions without having to acquire programming skills on top of their design skills. (Conversely, it means that programmers can employ their skills in a way that can be exploited by designers, rather than involving themselves in design.) Parameterized actions provided by an authoring system to be used in this manner are usually called *behaviours*. Since behaviours are going to be extensively re-used, considerable care must be taken in writing them, so that, for example, ECMAScript behaviours cope gracefully with the divergent object models and capabilities of different user agents.

What do we mean by describing behaviours as *parameterized* actions? Simply that each behaviour is an abstraction that captures a class of actions — such as displaying a message to a Web browser's status line — from which individual actions, such as writing the specific message `Please wait` to the status line, can be generated by providing a value for some variable, the behaviour's *parameter*. In this case, the parameter is the message to be displayed; in general, a behaviour may have several parameters. It follows that an authoring system that supports behaviours must provide some means for authors to provide parameter values, and be able to generate code by combining these values with a behaviour. They must also be able to attach the resulting actions to events chosen by the author.

The simplest approach is to build a limited set of behaviours into the browser. You might consider the link-following capabilities of all Web browsers as an instance of this approach. A slightly fuller set of behaviours is supported by PDF files. Recall that in Chapter 9 we described how to add links to a PDF file using Acrobat Exchange: after selecting an active area, a dialogue box appears, which previously we used to set up the parameters of a link. However, the pop-up menu labelled Type in that box (see Figure 9.3 on page 276) can be used to associate different actions with the active region. The possibilities include automatically executing a menu command, playing a QuickTime or AVI movie or a sound clip, launching a different program to open a file of some type other than PDF, or opening a file on the World Wide Web identified by a URL.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 490
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

The path names of the movie, sound clip, or file, and the URL are the parameters to these behaviours.

The repertoire of behaviours available in Acrobat is fixed, and the actions themselves are coded as part of the reading software. In contrast, modern Web browsers include code to interpret a scripting language, so that arbitrary actions can be written in that language. You should be able to see in outline how, instead of being written by hand like the examples in the previous section, ECMAScript actions can be parameterized as behaviours and incorporated into HTML semi-automatically. In order to fill in the details, we will describe the interfaces to behaviours provided by a representative HTML authoring program, Macromedia Dreamweaver, and sketch their implementation.

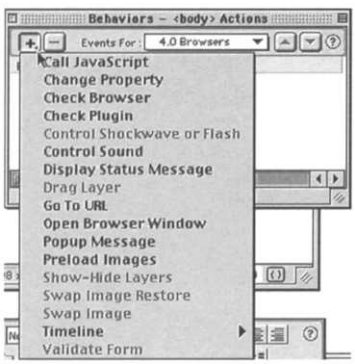


Figure 14.4
Dreamweaver's behaviours palette

Most HTML authoring programs (as against general-purpose text editors with HTML modes, such as Alpha or Emacs) use a similar interface, comprising a document window providing a WYSIWYG view of the document being composed, and several context-sensitive floating palettes, whose contents change according to the element selected in the document window. These palettes allow attributes of the selected element to be changed. Dreamweaver's behaviours palette (shown in Figure 14.4) is used to attach actions to events associated with the selected element. Clicking the + button causes a pop-up menu to appear, as shown, listing the names of all the available behaviours — Dreamweaver ships with a small collection, others can be added, as described later. Menu entries for behaviours that, for some reason, are not applicable to the currently selected element are disabled. For example, unless an image is selected the `Swap Image` behaviour will not be available. When a behaviour has been selected, a dialogue box is displayed, with fields for any parameters that are needed to generate the code. Once these have been supplied, the event to which the action is to provide a response is selected; each behaviour has a default, but any other event supported by the DOM can be chosen instead.

Since Dreamweaver is an HTML editor, its basic function is to change the contents of HTML documents. It can therefore embody the association of document element, event and action by writing suitable ECMAScript code in the document's `HEAD` to define a handler function, and then inserting an on*Event* attribute into the start tag of the selected element. For example, suppose an `IMG` element is selected, and two instances of the `Display Status Message` behaviour are attached to it, one for the event `MouseOver`,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 491
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

the other, with a blank message, for `MouseOut`, so that the first message will be displayed for as long as the cursor is over the image. Dreamweaver will insert a script equivalent to the following one into the document head:

```
<SCRIPT type="text/javascript">
function displayStatusMsg(msgStr) {
  status = msgStr;
}
</SCRIPT>
```

and decorate the `<IMG>` tag with attributes similar to the following:

```
onMouseOver=
    "displayStatusMsg('The cursor is inside the image')"
onMouseOut=
     "displayStatusMsg(' ')"
```

The collection of behaviours shipped with Dreamweaver provides an insight into the range of actions that can be readily added to a Web page in this way. (Looked at more cynically, since most of the other powerful HTML tools presently available support roughly the same set of actions, this collection provides an insight into Web page design clichés — at least until the next round of updates to these programs is released.)

These behaviours can be divided into groups. The first, consisting of `Display Status Message` and `Popup Message` display messages in response to events, in the status line and a dialogue box respectively. The second group consists of functions that check the browser's capabilities. `Check Browser` is used to determine which version of which browser is being used, and to select different URLs on the basis of this information, so that, for example, a user viewing a page with an old version of Navigator would be transferred to a simplified version of a page. `Check Plugin` can be used similarly to send users to different pages depending on whether they have installed a particular plug-in or not. For example, users without the QuickTime plug-in would be directed away from pages with embedded movies.

The next group of behaviours offers control over browsers and page elements. If a page has embedded Flash or Shockwave movies, or sounds, the `Control Shockwave or Flash` and `Control Sound` behaviours can be used to start or stop them, or for movies, rewind or go to a specific frame, all in response to any of the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 492
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

DOM events. For example, a sound could be made to play when a page is loaded, and stop when the user clicks on an image. The apparently redundant `Go To URL` is used to load a new Web page; the reason it is useful is that it understands about framesets, and can be used to change the contents of several frames with a single click, without the need for an intermediate frameset. `Open Browser Window` spawns a separate window to display a new page.[22]

22 Compare `show="new"` in SMIL.

Probably the most popular group is concerned with images and motion graphics. `Swap Image` provides the action needed to implement rollover buttons, or to perform image replacements in response to any chosen events. `Preload Images` performs the preloading operation described previously, to ensure that image swaps work smoothly. It is usually included automatically with `Swap Image`.

⇨ Image rollovers are such a common requirement for fashionable Web pages that Dreamweaver provides an `Insert Rollover` command, which allows you to create a rollover effect simply by selecting the images for the two states. Inserting one of them and creating the necessary code is all done in one step. Most HTML authoring packages provide this function, as do many graphics packages that have been augmented with Web capabilities or designed for preparing Web graphics. It can only be a matter of time before your emailer does too.

`Show-Hide Layers` controls the visibility of layers, and `Change Property` is a general-purpose behaviour for altering the style of a document element. The final behaviour in this group is `Drag Layer`, which can be used to add basic drag and drop functionality to absolutely positioned elements of a page, or to implement analogue-like controls, such as dials and sliders. It provides a powerful argument in favour of the use of behaviours, since implementing dragging within the confines of ECMAScript and the DOM requires some non-trivial programming: all mouse movements must be tracked and the position of the dragged object updated in response. Care must be taken to observe the physical constraints of the page. If you also want, for example, to reserve certain areas of an image as 'drag handles'; to provide a target for each dragged object, allowing it to snap to its destination when it is close enough; to support several draggable layers on the same page; and to work with as many versions of different browsers as possible, you would have to write quite a lot of code. As it is, it may be necessary to add hand-made code to the basic behaviour to monitor the position of the moving element.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 493
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

The behaviours that we have described are not built in to Dreamweaver in the same way that PDF behaviours are built in to Acrobat, new behaviours can be added to extend the repertoire. The mechanism by which this is achieved is an interesting exercise in economy, since all it requires is HTML and ECMAScript — which Dreamweaver has to be able to interpret already.

A behaviour is just an HTML file, whose `HEAD` contains some ECMAScript definitions, and whose `BODY` is a document that will be rendered in a dialogue box when a user selects this behaviour from the pop-up menu in the `Behaviours` palette. Usually, the document will contain a `FORM` element, with fields for accepting the values of the behaviour's parameters.[23] When the form is filled in, these values become available to scripts within the behaviour via objects in the same way as elements' attribute values.

The author of the behaviour must provide two functions: `behaviorFunction` and `applyBehavior`, which both return strings to be inserted in the HTML document being edited. The first returns the code for a handler function definition, the second a call to that handler, to be inserted into the selected element's start tag. Typically, the strings returned by both these functions will incorporate values elicited from the user by the form in the behaviour's `BODY`. Behaviours may include several other function definitions, notably `canAcceptBehavior`, which returns a list of events with which the behaviour can be associated, allowing Dreamweaver to correctly set up the pop-up menu by which the event is chosen when a behaviour is added to a document. A `displayHelp` function can also be defined, which returns a string describing the behaviour when the help button in the behaviour dialogue is pressed.

One can imagine simpler ways of implementing behaviours — using action templates, for example — but this approach permits great flexibility, since all the features of ECMAScript are available for creating the function definitions and calls. (If you know about JavaBeans, you might like to compare the approach with the way a programmer implements the `PropertyEditor` interface so that visual development environments can manipulate the properties of beans.)

23 We have not decribed HTML forms, since they are quite complex and not particularly interesting in the context of multimedia. Consult one of the HTML references for details on forms, the elements that may be used within them, and the special events that are associated with form elements.

Earlier, we showed how layers can be crudely animated using a script. The code for performing this type of animation is as amenable to prefabrication as the behaviours we have just seen, but a different interface is required for conveniently attaching the code to a document. The familiar timeline is provided by Dreamweaver and other authoring tools that support animated layers. Given their shared provenance, it is not surprising that Dreamweaver's timeline resembles a simplified version of Flash's. It appears in its own

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 494
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

palette, with the document view serving the same function as the stage in Flash. A layer can be dragged to the timeline, where a bar appears, representing its extent in time. This can be stretched or shrunk as necessary. Key frames can be added to the timeline at any point, and the layer can be moved to its desired position in that key frame. Movement between key frames is linearly interpolated by the script generated by Dreamweaver from the timeline.

➲ The code generated by Dreamweaver, although it is based on the same principle as the code we wrote to animate some text in the previous section — move the element to its new position and set a time out — is slightly smarter, because it precomputes all the necessary positions, and stores them in an array, which leads to more efficient animation.

Timeline Scripting and Behaviours

Scripting can, as we have shown, free hypermedia from its purely spatial model, and add an element of temporal organization to it. Where multimedia is primarily organized on a timeline, scripting can provide interactivity and non-linearity. Timelines provide new sites for actions — an action can be attached to a frame and triggered when playback reaches that frame — and new sorts of action, ones that control playback. These two innovations offer three new opportunities to multimedia authors. By attaching playback control actions to static elements — such as buttons — that can respond to events, timeline controls — such as fast forward, rewind, and pause buttons — can be implemented. By attaching other actions to frames, their effects can be choreographed in time. By attaching timeline-controlling actions to frames, the timeline can be made to loop. Notice that the effect of actions attached to frames is fully determined — only elements that can respond to events initiated by users can produce non-linear behaviour.

We shall begin by considering Flash, which, although it is primarily an animation tool, provides some simple behaviours. Actions generated by supplying parameters for these can be attached to key frames (not interpolated frames) or to special symbols called *buttons* (irrespective of their appearance). An action is attached to a key frame simply by selecting the key frame and the appropriate menu command, choosing one of the available behaviours, and

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 495
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

entering values for its parameters. The action is triggered when playback reaches the frame to which it is attached. Attaching actions to buttons is achieved by a similar sequence of operations; first, however, the button must be created. Flash buttons are special symbols (in the specialized sense in which Flash uses the word), consisting of four individual frames, the first three of which correspond to the button's appearance when it is in one of the states 'up' (the cursor is not over it), 'over' (it is), or 'down' (the mouse is clicked while the cursor is over the button). The fourth frame defines the 'hit area', the region that responds to the mouse. Once a button has been created, it can be placed on the stage like any other symbol, animated, and have actions attached to it.

The behaviours Flash offers to control timelines are elementary: `Play`, `Stop`, and `Go To`. The last of these takes a frame number or a label — which can be attached to a frame as a mnemonic — as its parameter; it also has options for selecting the next or previous frame as the destination. Whatever the destination, a further option to the `Go To` behaviour can be used to determine whether the movie should play from the new frame or stop there. It should be evident how these behaviours can be used with buttons to start and stop a movie or jump to a selected point. They are of limited use when they are attached to the timeline; using a `Go To` action to produce a loop is the only effective option. Combining a looped movie with buttons that cause a jump to a selected frame is, however, sufficient to provide a selection of different interfaces for offering a user some meaningful control over a presentation.

A straightforward example of such a combination allows us to provide a hypermedia-like interface. Suppose a movie consists of several distinct scenes,[24] each one associated with a particular subject. For example, if we wished to demonstrate the effects that could be achieved using Flash's built-in drawing tools, we might produce a set of short animations, each of which showed a picture being built up using one of the tools: we could do some motion painting with the brush tool, move some shapes about with the arrow tool, and so on. We want to present the user with a welcome screen containing icons of each of the tools, so that clicking on a tool causes the corresponding animated demonstration to play, after which the welcome screen returns (see Figure 14.5).

This is easily done. The welcome screen can be a single frame — let's say the first frame, though it doesn't have to be. Attached to this frame is a `Go To` action, with the frame itself as the destination;

24 In fact, Flash allows you to split movies into separate named scenes that can be treated as distinct entities.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 496
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

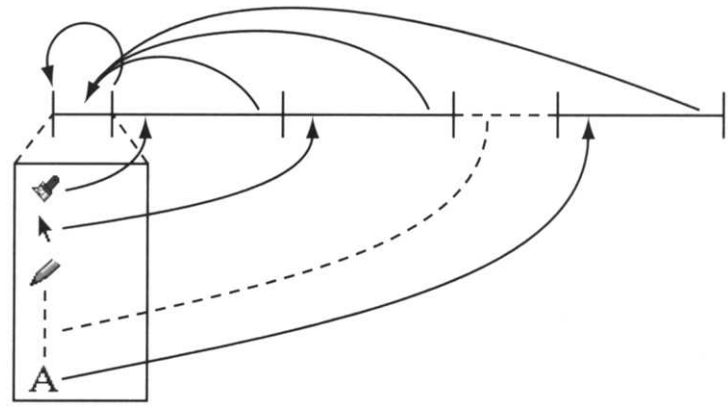


Figure 14.5
Structuring a timeline with selections

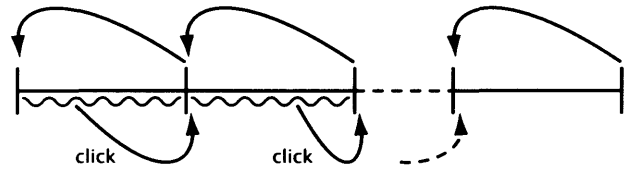


Figure 14.6
Structuring a timeline as levels

i.e. the movie will hold on this frame forever, unless some action causes it to do otherwise. Each of the icons in this frame will be a button with a `Go To` attached, with the starting frame of the corresponding demonstration as the destination. The final frame of each demonstration will have `Go To Frame 1` attached, to ensure that the welcome screen returns after it has finished. To complete the presentation, an exit button with a `Stop` action should be added to the first frame, to give the user a way out of the infinite loop.

Because buttons in Flash are symbols, they can be animated, which allows for a type of interface that is not available in hypermedia (without the aid of scripting to do the animation). A slightly fanciful example will demonstrate this. A little exercise for helping new computer users to become proficient in using the mouse might be organized as a series of levels of increasing difficulty. At each level, the task is to click on a moving object. When this is done succesfully, the user moves on to the next level, where the object is smaller and moving faster in a more complicated path, possibly with extra objects moving around and getting in the way (see Figure 14.6).

The scenario more or less determines the implementation. Each level is a short animation, with a `Go To` action attached to its

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 497
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

final frame, taking it back to its first frame. These animations are constructed separately, each with its own object moving on its path, so that its position in the final frame is placed to join up with the position in the first. The objects are Flash buttons, with an action that goes to the first frame of the next level. Since actions are triggered by mouse clicks, all that the user needs to do to advance to the next level is catch the object with the mouse. After a suitable number of levels, the action on the object at the hardest level takes the user to a short congratulatory animation, after which they can get on with using their mouse in earnest. We could (and probably should) have added a facility for going back a level, or giving up, and possibly a way of selecting a starting point.

As well as demonstrating the power of behaviours — putting together a prototype took us hardly any time — this example shows some of their inadequacy. The mouse trainer would be more useful if it could take account of how long a user took to master a level, by counting the number of iterations of the loop before they caught the object, so that if a user was obviously having difficulty they could be taken on at a slower pace than those whose eye and hand coordination was well developed. This cannot be done, since Flash provides no mechanism for storing state information. A scripting language could be used to do this[25] — but even a scripting language could not be used to save the state of a movie so that it could be restarted at a later date from the same point, unless a facility to do so was included as part of the object model. Generally, this is not done — multimedia scripting languages do not usually provide any facilities for writing to disk.

Flash's most advanced behaviour is designated `Tell Target`, which is used to send an event from one object to another. This is commonly employed to enable buttons to control individual elements of a movie. You will recall that a Flash movie can be included in another movie, in the form of a symbol. The included movie has its own timeline, which can be controlled by actions. `Tell Target` enables a different symbol to trigger one of those actions. To make identifying the target easier, symbols can be given names. Suppose, as a simple example, we have a movie consisting of nine frames, containing the digits 1 to 9. A `Stop` action attached to the first frame prevents the movie from playing. We now construct a single frame movie, by placing an instance of this movie on the stage, giving it the name `counter`, together with two other buttons, one representing a + sign, the other a - sign. We attach

25 Version 4 of Flash added a primitive scripting language. Unfortunately, it uses its own idiosyncratic syntax, instead of being based on ECMAScript.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 498
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

to the + button a `Tell Target` action, with its target parameter set to `counter` and the action for the target to perform specified as `Go To Next Frame`; a similar action is attached to the - button, telling `counter` to go to the previous frame. When this movie is played, nothing happens spontaneously — there is only one frame — but if the + button is pressed, the displayed digit is incremented, provided it is less than 9, and if the - button is pressed, the digit is decremented if it is greater than 1.

You should be able to see how `Tell Target` actions can be used to implement controls that affect individual elements of a movie. Again, the restricted nature of the available behaviours imposes some limitations on what can be achieved. For instance, it is not possible for the + to disable itself when the displayed count reaches 9. However, with ingenuity, complex controls such as pop-up menus can be assembled out of buttons and movies, using the built-in behaviours.

When we consider how techniques similar to those we have described in the preceding paragraphs can be applied in the context of the World Wide Web, we find a somewhat different situation. Timeline animation on a Web page is itself implemented using scripting, as we have previously described, so actions that are to be triggered when an animation reaches a certain frame can simply be embedded in the function that performs the animation, to be executed conditionally when some current frame counter reaches the appropriate value. For example, if we wished to modify the 'motion blur' example from page 464 so that the colour of the moving word `MOTION` changed to fuchsia at the fifteenth frame, we could do so very easily, by adding a new global variable

```
var action_frame = 15;
```

and inserting two lines of extra code in the function `move_the_text()`:

```
if (--action_frame == 0)
    document.all.layer1.style.color = 'fuchsia';
```

If you compare these two lines of code with the two lines that move the text layer in the same function, it is easy to appreciate that in the context of Web pages, animation is just a special case of time-based execution of scripting code. Conceptually, though, it may often be helpful to treat the timeline as an organizing framework, just as we do in an intrinsically time-based system such as Flash. In that case, what is required is a reusable framework for arranging actions,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 499
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

including animation, in time. If this can be done, the timeline can be presented as part of the interface to an HTML authoring system. The three opportunities identified at the beginning of this section as being offered by the combination of actions and timelines become available to Web designers.

We briefly described at the end of the last section how Dreamweaver does offer such a timeline interface, which it uses to generate scripts using timeouts to implement animated layers. The facilities go further: the timeline has a behaviour channel, which allows actions to be attached to any frame. Arbitrary ECMAScript code can be used, as can any of the available behaviours. Several different layers can be controlled by the same timeline, just as several different symbols can be animated in Flash. In addition, several different timelines, each with its own name, can be attached to the same page. This enables different animation sequences to be controlled independently by events. The behaviours `Play Timeline`, `Stop Timeline`, and `Go To Timeline Frame` are all parameterized in a timeline's name, as well as being associated with events and document elements in the usual way. It is thus possible to provide a set of controls for playing an animation by associating these behaviours with images and binding them to the animation's timeline. Multiple sets of controls, controlling multiple animations on the same page, then follow naturally from the ability to define several timelines for the same document.

Because putting together a reusable scripting framework that supports time-based actions, interpolation and key frames is not a trivial exercise, it is usually advisable to adopt an existing one if it is available. It may then sometimes be necessary to mix hand-crafted scripting with prefabricated behaviours to achieve the desired effect — since a fully-featured scripting language is available, effects that are not possible within a fixed repertoire of actions can be implemented. For example, suppose we wanted to re-work our earlier Flash example of a counter controlled by two buttons as a scripted Web page. We could place GIF images for the two buttons and the first digit each on their own layer so that we could position them where we wanted. By dragging the digit's layer on to the timeline palette, creating nine key frames and resetting the `src` attribute of the image in each key frame we could create an animation that ran through the nine images of digits. By not attaching any `Play Timeline` action we would ensure that, as before, no change took place spontaneously. Adding a `Preload`

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 500
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Images action to the document's HEAD will ensure that when we do want the digits to change they do so smoothly.

All we need to do to complete the example is add actions to the + and - images — which are serving as controls — to cause the animation to go to its next and previous frame respectively, in order to achieve an effect identical to that of our earlier effort with Flash. Dreamweaver only ships with a behaviour that causes a timeline to go to a specific numbered frame, not to the next or previous one. By inserting such an action and inspecting the generated code, we discover that Dreamweaver uses a function called MM_timelineGoto, which takes the name of a timeline as its first argument, and a frame number as its second, to implement its Go To Timeline Frame behaviour. Armed with this knowledge, we can not only implement the required operations, we can make something sensible happen if the digit's value goes outside the range one to nine — we choose to make it wrap round.

We begin by declaring a variable, and asserting that, except while an action is being performed, its numerical value will be equal to that denoted by the displayed digit. To establish this invariant we must initialize the variable to one.

```
var current_digit = 1;
```

Considering the layer holding the digit's image element as an animation, our assertion also amounts to saying that this variable's value is equal to the current frame number. We can therefore use it in a pair of functions that cause the next and previous frame to be displayed by incrementing or decrementing its value and then passing it to MM_timelineGoto as its second argument. If current_digit is about to go out of range we first make a suitable adjustment to produce the intended effect.

```
function up()
{
  if (current_digit == 9) current_digit = 0;
  MM_timelineGoto('Timeline1', ++current_digit);
}

function down()
{
  if (current_digit == 1) current_digit = 10;
  MM_timelineGoto('Timeline1', --current_digit);
}
```

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 501
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

(`Timeline1` is the name of the timeline corresponding to the sequence of digits.) The functions `up` and `down` can be used as handlers for a `mouseDown` event in each of the two control images.

Combining behaviours, timelines and *ad hoc* scripts within an HTML authoring environment is a relatively easy way of producing dynamic Web pages that exhibit change with time and respond to user input in a variety of ways. It is always the case, though, that what can be achieved is restricted by the limitations imposed on Web scripting languages for the sake of security.

The combination of scripts with a timeline is taken to the extreme (and possibly beyond) in Macromedia Director. To begin with, Director provides behaviours that can be attached to frames or sprites to implement much the same range of time-based actions and controls as Flash, but in the richer multimedia environment provided by Director. A `Message Sprite` behaviour can be used like `Tell Target` in Flash to control one sprite from another. At the next level of sophistication, Director allows authors to compose their own behaviours by combining actions — chosen from a built-in set — into a sequence, which is then treated as a single behaviour. Composition is done simply by repeated selection from a menu, so no real scripting skill is required. The set of available actions includes several to control playback, such as `Go to Frame` and `Wait on Current Frame`, others to make global changes to the playback environment, such as `Set Volume` and `Change Palette`, as well as some to control sprites and cast members: `Play Cast Member` plays a sound from among the cast, and `Change Cast Member`, which takes a sprite as one of its parameters, associates a new cast member with that sprite, so that its appearance changes. Behaviours attached to a sprite are triggered by the usual user-initiated events: a key or mouse button being pressed or released, the cursor entering, leaving, or being over the sprite. Those attached to a frame may be triggered when the playback head enters the frame or when it leaves it; an additonal event, `Prepare Frame`, which occurs between the playback head's leaving the previous frame and entering the current one, can be used to provide an even finer degree of control over the precise sequencing of actions.

The full range of Director — the thing that made it the leading multimedia authoring environment of its time — depends on the use of its scripting language *Lingo*. Lingo belongs to the tradition of 'English-like' scripting languages rather than those, such as ECMAScript, derived from C and other mainstream programming

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 502
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

languages. To anyone used to more established languages, the most noticeable aspect of Lingo is its verbosity: a property `p` of an object `obj` is referred to as `the p of obj`, for example, giving rise to expressions like `the rect of the member of sprite 19`; assignment uses the `set` keyword, in the style of Basic, as in `set name = "Wilfred"`.[26] At a sightly deeper level, Lingo ostensibly has no user-defined functions or subroutines. Instead, users can define handlers for 'custom events' — events that aren't predefined. In fact, custom handlers are functions, and custom events are calls to them. Handlers can take arguments and return values. For example, a handler might be defined in the following way:

```
on mean a, b
    set m = (a + b)/2
    return m
end mean
```

and invoked as follows:

```
set average = mean(843, 438)
```

In the case of built-in events, the syntax used for the handler's definition conflates it with the association with an event; using the same syntax for user-defined functions, and calling them custom handlers, is merely a piece of conceptual economy, with no deep semantic significance.

Underneath the cosmetic differences, Lingo has roughly the same facilities and expressive power as ECMAScript, which it also resembles in being untyped. It may have some claim to be object-oriented, rather than object-based,[27] since it has constructs that behave somewhat like classes and inheritance, though again unconventional terminology is used to describe it ('parent scripts' for classes, and 'ancestor scripts' for base classes). The object model is not separated from the language, as it is in World Wide Web scripting, but there is clearly an object model embedded in Lingo, which encompasses all the elements of a Director movie — score, cast, sprites, and so on. Lingo therefore provides a completely flexible way of controlling all those elements on the basis of user input and computation carried out by scripts.

Not only do Lingo scripts offer extra flexibility, compared with behaviours, they can be attached to the production in more places, and be triggered by some extra events. Scripts are divided into different categories, depending on which elements of the

26 With the release of Director 7, some changes were made to Lingo, so that its syntax is beginning to converge with ECMAScript. Object properties can now be accessed using a dot operator, and `set` is no longer required. (On the other hand, `to` can, and always could, be used instead of `=` in an assignment.) The old syntax is retained for compatibility.

27 In the absence of static type checking, there is little significant difference between the two approaches to objects.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 503
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

production they are attached to. 'Score scripts' are the same as behaviours, and may be attached to sprites or frames. These scripts can respond to mouse, key and frame-related events, with the sprite's handler taking precedence over the frame's if there is any conflict. A sprite may also respond to `beginSprite` and `endSprite`, which occur in the first and last frame, respectively, of its duration. 'Scripts of cast members' are attached, as the name suggests, to a cast member itself; remember that a cast member is a media object, such as an image or a sound, whereas a sprite is an instance of that object, placed on the stage, with a specific duration. These scripts respond to the same events as frame scripts, although the latter take precedence. Every sprite that uses a cast member automatically acquires the scripts of that cast member. Moving further outwards in the structure of a production, 'movie scripts' are attached to an entire movie, and 'primary event handlers' to the entire production (which may comprise several nested movies). Movie scripts respond to three additional events: `prepareMovie` (the very first event that occurs), `startMovie` and `stopMovie`. Primary event handlers can only be set up for key presses, mouse clicks, and timeouts.

The final category of scripts is, in some ways, the most interesting. 'Parent scripts' contain the definitions of a collection of handlers. An object may be constructed from a parent script using the `new` statement, which causes an object to be created, and attaches all the handlers in the parent script to the object. Parent scripts usually include a special handler for the `new` event, which is invoked when an object is created, and used to initialize it. If you are familiar with object-oriented programming, you will recognize the `new` handler as a constructor, and the handlers in the parent scripts as methods belonging to a class which the parent script defines. Parent scripts may also include some property definitions; any object created from the script acquires the same properties. (We might call them 'instance variables'.) An object created from a parent script is called a 'child object' (rather confusingly, since inheritance is not yet involved,): it is just an object in the classical sense of some code and some data structured so that it responds to certain messages. Every child object created from a particular parent script responds to the same set of events in the same way; the values of each child object's properties may change independently, so that each child object is an instance of the particular 'sort of thing' that its parent script implements. If a parent script has a sprite as one of its properties, it can be used to create generically similar sprites that respond to the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 504
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

same set of events. This is clearly more manageable than creating sprites and attaching the same scripts to each one individually. A parent script may declare another parent script to be its ancestor, meaning that any events that cannot be handled in the original parent script will be passed on to the ancestor — in other words, the parent script *inherits* handlers from its ancestor. Ancestors may have ancestors of their own, so inheritance hierarchies can be constructed.

We have seen that custom handlers can be called from any script. The properties of sprites, such as their location, size, cast member, and so on, can be changed within handlers. In other words, a sprite can be controlled by scripts. Together with the structuring facilities provided by parent scripts, this offers the possibility of an alternative to the timeline for organizing movies: organize them as programs. For example, instead of implementing a bouncing ball by adding key frames to the timeline at the extremities of its motion, and interpolating a sequence of frames in between, it could be implemented using a single looped frame, with a handler triggered each time the frame was entered that computed a new position for the ball sprite and updated its location accordingly.

The advantage of this way of controlling the elements of a movie is that scripts can perform arbitrary computation. In the example just cited, the ball's position could be calculated from equations derived from the laws of motion governing its trajectory, thus producing a more realistic bouncing movement than that which would be produced by interpolation[28] between the extremities of the motion. Parameters such as the ball's elasticity could easily be changed by editing the script. In more elaborate productions, scripts can be used to endow sprites with artificial intelligence, enabling them to react to user input, or to each other, in less predictable and possibly more satisfying ways than the simple branching behaviour that is the natural concomitant of timeline organization.

28 Director uses Bézier interpolation by default.

⇨ There have been attempts to create multimedia authoring environments that started with the interactions between objects as their basic organizing principle. Such object-oriented environments, as they are inevitably called, have failed to achieve widespread popularity. Despite an enthusiastic cult following, the most succesful such system, Digital Chisel's mTropolis (sic), was killed off shortly after being acquired by Quark, leaving no serious object-oriented authoring environments at the time of writing.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 505
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Beyond Scripting

Traditionally, programmers have thought in terms of creating an application that does something, and adding a user interface to it. The more enlightened programmers will have designed the interface as an integral part of a complete system, and will probably have used some application framework or set of reusable components to build the interface, but the implicit model of program structure is one in which the user interface is a subsidiary component of a system intended to perform some task. Where we need a multimedia interface, or to manipulate media objects, it may be more appropriate to think about creating the user interface in an authoring environment and adding the functionality inside it, so that the interface becomes the organizing principle, with the code to perform the computational task added to the interface instead of the opposite way round. The rôle of scripting, though, has traditionally been seen as one of controlling the elements of a production and reacting to input, and scripting languages usually lack features, such as file input and output, that are necessary for many applications. (ECMAScript on the Web, as we have seen, must lack such features, for security reasons.) Hence, while the scripting facilities and behaviours we have described so far can do plenty of things to make the displayed media elements respond to the user, they are really only enhancing the user interface, not adding real functionality. To do that, some additional mechanism is required.

One approach is exemplified by Director. *Xtras* are code modules, written in C or C++, which can be used to extend Director, adding capabilities that the basic application does not provide. These can range from support for new media types to networking and database management.

There are four types of Xtra. Transition Xtras provide new forms of transition, in addition to the built-in dissolves, wipes, and so on; tool Xtras add new functions to the authoring environment. The remaining two kinds are the most germane to our present discussion. Sprite Xtras are used to add new classes of objects to the range of cast members. A sprite Xtra usually defines some visible representation, and a set of properties and behaviours, so sprites created from it can handle events and perform computation. The difference between sprite Xtras and parent scripts is that the former, being written in C, are not subject to any restrictions on the kind of

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 506
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

operations they can perform. Sprite Xtras can be written to interact with databases or other persistent data, or to represent and display elements of a complex simulation, for example. The final type of Xtra is the scripting Xtra, which adds new commands to Lingo itself. An example is the `fileio` extra, which provides commands such as `open` and `write` for accessing files; these commands are not otherwise available in Lingo.

Xtras must be written to conform to an interface specification that allows Director to recognize them and manipulate them like elements of its own environment. Director prescribes the interface — the function names and their signatures — and the Xtra programmer provides the implementations. Every Xtra must implement a function that registers it, passing various identifiers and information about the Xtra's capabilities to Director, so that it can be manipulated in the authoring environment. For example, its name will be added to appropriate menus, and, if it implements a custom editor for changing its properties, this will be made available when it is required. Additionally, a sprite Xtra must implement functions to create instances of itself, draw a representation of itself, if one is needed, and respond to events; a scripting Xtra must provide a despatching function that allows it to be called from Lingo. Behind these interfaces, Xtras can perform any computation at all, so that, as we suggested before, an arbitrarily complex multimedia application can be constructed by embedding computational elements, in the form of Xtras, inside a multimedia production made in Director.

⇨ Besides adding new functionality, there is another reason for resorting to Xtras and similar extension mechanisms that allow programmers to mix C++ or a similar language with scripting. At the end of the previous section, we mentioned the possibility of using scripts to endow sprites with some artifical intelligence. One could do this in Lingo. However, when things have got as far as artificial intelligence, many programmers would doubt the wisdom of continuing to use a scripting language instead of a programming language that offers more sophisticated scoping, compile-time type checking, and the object-oriented facilities that can be built on them. Lingo was designed to be accessible to non-programmers, but using it in advanced ways is programming, and anyone who can program in Lingo will be able to program in Java or C++, and do so more effectively with the assistance of language features designed to help in the task of structuring complex programs.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 507
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

The most loudly trumpeted mechanism for extending multimedia productions through conventional programming must be Java *applets* — small programs, written in Java, that can be embedded in HTML documents. With the advent of applets, executable programs became, in effect, another media type that can be included in Web pages. Java is a complete object-oriented programming language, and it has a set of powerful libraries that allow applets to display graphics and animation, play sounds and movies and handle events. Subject to security restrictions described below, they may also communicate over networks, access databases, control domestic appliances, and carry out many other computational tasks.[29] The technology required to execute applets comprises two components: a way of embedding applets in HTML, and a way of executing code from within a Web browser.

29 Applets are not the only programs that can be written in Java. Standalone applications that do not depend upon a Web browser for their execution can also be generated.

Originally, the first was provided by a special-purpose APPLET document element, but, in accordance with the view that an applet is a media element like a QuickTime movie, APPLET is deprecated in HTML 4.0 in favour of the OBJECT element, which we introduced in Chapter 13. For executable OBJECTs, the `classid` attribute is used to specify a URL where the implementation can be found. Any OBJECT can contain PARAM elements, as we described in conjunction with using OBJECTs to embed movies in HTML documents. For applets, these can be used to pass named parameters to the applet. For example,

```
<OBJECT classid="NervousText.class">
<PARAM name=text value="Hooray for Captain Spaulding!">
The most famous applet should be executed here.
</OBJECT>
```

will embed an applet that runs the notorious 'nervous text' demo, with the string `"Hooray for Captain Spaulding!"` being passed to it as the value of its `text` parameter.

An applet OBJECT can be formatted and positioned using CSS, like any other document element. (An OBJECT is an inline element, like an IMG.) The height, width, and co-ordinates, if specified, of an OBJECT define an area within the browser window into which the applet can draw and write, treating it as a Java frame. Applets can also create their own windows as they run, but these always carry an intimidating warning in their title bar, so that downloaded applets cannot masquerade as system windows, and elicit information to which they have no right from unsuspecting users.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 508
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Applets do not actually have to be written in Java, although they usually are. The `classid` attribute's name seems to embody the assumption that applets are Java classes, but, providing a suitable method for executing the code is implemented in the browser, Java does not have a monopoly on applets. The `OBJECT` element has an attribute `codetype` that allows the type of executable code to be specified, just as its `type` attribute allows the type of data to be specified.

Java class files only store the code for a single class; in the case of an applet, the class that implements it, which must extend `applet`. If an applet uses other classes, and perhaps some data files that are always the same, you can package everything it needs into a single archive, and use the `archive` attribute to inform the browser of its whereabouts, so that the entire archive can be retrieved at once, without having to make multiple connections to get all the pieces. You still need the `classid` attribute, to identify the applet from among the classes in the archive.

If you need to preserve the state of an applet between invocations, you can use Java's object serialization facility, and specify its serialized form in the <OBJECT> tag. In this case, you must use the `data` attribute, not `classid`.

One of Java's best publicized features is its *platform independence*, often summed up in the slogan 'Write once, run anywhere'. This is clearly a necessity if applets are to be able to run in any browser, on any machine, under any operating system. Equally clearly, platform independence cannot be achieved by a conventional compiler that generates machine code for a specific machine. Java's solution is to compile the source code into code for a *virtual machine*. A virtual machine is a fully specified machine, with its own instruction set, data types, and so on, but it does not exist in hardware. Its instructions are interpreted by a piece of software — an *interpreter* — which emulates the behaviour of the virtual machine. Interpreters for Java's virtual machine code have been written for all major platforms. Dissatisfaction with the performance of early interpreters has led to the adoption of more efficient techniques. In particular, 'just-in-time' compilers are used to generate native machine code on the fly from the virtual machine code, when doing so would provide better overall performance.

If you have read some of the more enthusiastic accounts of Java, you may be forgiven for thinking that the technique of compiling to a virtual machine code which can be interpreted on a range of host machines is one of its unique innovations. In fact, the technique is

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 509
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

an old one, by the standards of computing, dating back at least as far as the 1960s.

To execute an applet, a Web browser must first download the virtual machine code generated from the Java source, and then invoke the interpreter on the local machine to run it. The Web browser sets up the appropriate graphical context, on the basis of the `OBJECT` tag and any stylesheet applied to it, and performs security checks on the applet's code to ensure that it cannot compromise the client machine.

Security is a subject that has been given great consideration in the design and implementation of Java. A description of how the security features are implemented is beyond the scope of this book, but their effect is that applets can be run with different levels of privilege, so that applets downloaded from trusted sources can be permitted to perform operations that are denied to those whose provenance is dubious. In particular, applets that are downloaded from the World Wide Web are usually run under the same restrictions as are applied to scripts embedded in HTML. They may not read or write local files, run programs on the client machine, determine any but the most general information about the machine they are running on, or open network connections to any other machine than the host from which they are downloaded. Applets running under these restrictions are colloquially said to be running in the 'sandbox' (like children playing harmlessly). These restrictions mean that applets running in the sandbox are not much use, but they are absolutely necessary, given the completely open nature of the Internet. The sandbox restrictions are unnecessarily severe for applets that are downloaded from a server on an organization's internal local area network, though. There are mechanisms for determining whether an applet comes from where it purports to, and whether it has been tampered with. This means that it is possible to verify that local applets really are local, and should be allowed out of the sandbox to do useful computation. Unfortunately, with each successive release of Java, well publicized bugs have been discovered in the implementation of the security features, leading to a decline in confidence in their reliability.

By now you may be wondering whether there is any point to applets, since, inside the sandbox, there isn't really anything you can do with an applet that you can't do with a script, stylesheet or plug-in, and outside it you could use an ordinary program. It is certainly true that the case for using Java as a means of enhancing the

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 510
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

interface to Web pages is much less compelling now than it was when Java first appeared, since, at that time, there was no client-side scripting, multimedia plug-ins were not widely available, and Web designers had little or no control over typography. Many early applets were intended to overcome these limitations, with animation being particularly popular (and particularly bad). Applets were also the only means of providing interactive Web pages. With these functions now implemented more accessibly by scripts and plug-ins, Java has had to work harder to justify itself.[30]

30
On the Web, that is. Java applications are subject to different criteria. Their portability and the scope of the available libraries remain significant advantages.

One argument in favour of applets is that they can perform arbitrary computation, and display graphics and other media, without the need for any plug-in. Thus, they can add new functionality to a Web browser, using no resources beyond the Java Virtual Machine interpreter provided on the host machine. Theoretically, this argument has some force, but in practice browser vendors seem to prefer to add extra functionality to their programs, and users seem content to tolerate the resulting heavy memory demands. Another occasional advantage is that, whereas the source of a script is physically part of a Web page and therefore available for all to see, an applet is download in binary form, so that its workings can be kept secret.

Generally, though, applets, like Xtras, only become worthwhile where complex computation must be carried out and its result integrated dynamically into a multimedia production. Most of the time, useful computation requires cooperation between the client (Web browser) and a server. As we have tried to show, server-side computation alone cannot do much more than enhance the user interface, and this can be done easily and adequately using scripts, as we described earlier. The subject of server-side computation, and communication between server and client will be examined in Chapter 15.

⇨ Some of Java's more vociferous supporters have advocated the use of applets as a means of software distribution — replacing conventional installers — which adumbrates new marketing models, and new types of computer. The idea is that applets will be kept on servers, either a central repository serving all the machines connected to an organization's local area network, or a software vendor's server on the Internet. Anyone wishing to perform some task on their computer will download a suitable applet. If data is also kept on a server, as it usually is within organizations, and conceivably could be on the Internet, the local computer only needs

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 511
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

to be able to run a Java interpreter; it does not require local storage (apart, perhaps, from a cache) and can be slimmed down to the bare minimum — a *network computer.* Since users do not keep copies of applications, they can be charged for use, not ownership. The use of trusted servers to hold applets that need to get out of the sandbox, and a solid implementation of the Java security model, are necessary if this mode of distribution is to be useful.

Enthusiasm for network computers peaked around 1997. Within organizations, the concept may yet prove viable, but domestic users currently show no signs of enthusiasm for abandoning their local disks in favour of hiring space on a server, or for downloading an email program every time they need to send a message. However, services such as 'video on demand' may be implemented using similar ideas.

The mechanisms we have described allow computation to be embedded inside a multimedia production, but sometimes it is necessary to write code that manipulates media data directly — somebody has to write the multimedia authoring programs, Web browsers and plug-ins, if nothing else. This does not mean that everything must be done from scratch, though. Most media formats can be manipulated using the objects or functions supplied by an accompanying library and its application programming interface (API). The QuickTime API, for example, which has interfaces to C (or C++) or Java, and has been integrated into Perl[31] includes a wealth of functions for playing, editing, and obtaining information about movies. A simple movie player, with cut and paste editing functions, can be implemented in a few pages of C, much of which is concerned with initialization, event handling, and window management. For Java programmers, the Java 2D and Java 3D APIs provide classes for creating and manipulating two- and three-dimensional graphics, while Java Sound provides audio and MIDI capabilities. The Java Media Framework is a collection of classes that support the capture and playback of synchronized time-based media, both from disk and as streams, and multimedia conferencing. While multimedia programming cannot be simple, APIs such as these enable it to be done at a level of abstraction that conceals much of the problematic detail, allowing multimedia to be relatively easily incorporated into many types of application.

31 MacPerl only.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 512
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Further Information

[Fla97] provides a full and readable description of Web scripting. [ECM97] is the formal definition of the ECMAScript language, and [DOMa] and [DOMb] define the W3C DOM. [Sma99] is an interesting book about Lingo scripting, concentrating on its object-oriented aspects. If you are a Java expert, [SWDB98] will tell you about the Java Media Framework.

Exercises

Note: Most of these exercises require some scripting ability. If you do not feel confident about writing scripts, see how far you can get using behaviours in an authoring program such as Dreamweaver.

1. Modify the 'Why a duck?' script on page 457 so that it checks for an undefined return value and takes some appropriate action.
2. Why does the HEAD element of an HTML document never have any event handlers attached to it?
3. Construct an HTML page containing two square buttons. Add event handlers, so that, when the cursor is over one of the buttons, the other one changes into a circle.
4. Construct an HTML page that behaves like an animation player. That is, there should be buttons to start, stop, and single-step through an animation. Add event handlers to make these buttons play a simple animation, held as a sequence of individual GIF files on the server.
5. Using CSS, construct a page of HTML that uses complex styling — special fonts, colours, and so on. Add a button to the page which, when clicked, will cause all the styling to be ignored, and the page to be displayed in the browser's default style.
6. How would you make text tumble across the screen in a Web browser?
7. Create an image map in which clickable areas will only be active on, after, or between certain dates (rather like a digital

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 513
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Advent Calendar, for example). When the clickable areas are activated, a thumbnail image is revealed which may be blown up to full size.

8. Suppose a Web site contains a large number of similar pages, each containing details of a feature film. Implement an interface to this site, using only client-side scripting, that provides the following: a welcome page contains two buttons, one marked 'view by date', the other marked 'view by director'. All the pages for individual films have a button marked 'next' and one marked 'previous'. If the user clicks the 'view by date' button on the welcome page, the page for the earliest film is loaded, and, henceforth the 'next' button takes them to the following film in chronological order, while 'previous' takes them back chronologically. If, however, they click 'view by director', the films are traversed in alphabetical order of their director's name. Assume that the collection of films whose details are recorded on the site is fixed.

9. Implement the mouse training game described on page 474 in Flash. Add the 'go back a level' and 'give up' functions mentioned in the text.

10. Redo exercise 3 as a Flash movie.

11. If you have access to Director, make a movie of a bouncing ball, implemented as suggested on page 482, as a looped one-frame movie. Use the equations of motion to ensure that the ball behaves in accordance with physics (as far as your knowledge of mechanics goes).

12. Give an example of a task that could be performed using a Java applet but not using client-side scripting.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 514
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

15 Multimedia and Networks

The relationship between computer networks and multimedia is a contradictory one: they are incompatible perfect partners. As the preceding chapters have shown, multimedia places considerable demands on resources: files are often very large, complex processing, such as decompression, may be required, response times must be short, and tight, sometimes complex, synchronization constraints must be respected. Networks are particularly poorly placed to satisfy these demands, because of the inherent limitations of present day technology, and because of the complex and unpredictable effects of the interactions between network components and patterns of network traffic. At the same time, it makes a great deal of sense to maintain multimedia data in a central repository, to be accessed over a network. Wasteful duplication of those same large files is avoided, there is no need for any physical medium, such as a CD-ROM, to be used for distribution, and the approach fits in with the current trend towards distributed systems of desktop machines connected to central servers. Furthermore, the combination of networks and multimedia makes new applications, such as video conferencing and live multimedia presentations, possible.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 515
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

The considerations in favour of delivering multimedia over networks are generally held to be so compelling as to make it worthwhile living with and trying to overcome the limitations.

Established multimedia authoring environments have been extended to support delivery over networks, especially by adding features to integrate them into the World Wide Web. For example, Director movies can be transformed into *Shockwave* movies, a compressed format which retains all of the media and interactivity of a Director movie, but is optimized for delivery over the Internet. A Shockwave plug-in for Web browsers is freely available, so that all the functionality of Director movies can be provided inside a browser. Some extensions to Director's scripting language Lingo have been made to support URLs and the fetching of data over the network, and Shockwave movies can interact with the browser through scripting and the document object model. This departure from the original model of offline delivery that served Director well during its early years is evidence of a recognition of the growing importance of online delivery of multimedia.

In this chapter we will take a closer look at the mechanisms, which we have so far only glanced at, that are provided to make it possible to deliver multimedia over a network. This material is among the most technical in the book, and you probably won't find it of much interest unless you have some knowledge of how computer systems and networks operate. Readers without this technical background may prefer to skim the opening paragraphs of each section to get some idea of what is involved in meeting the demands of distributed multimedia. Computer scientists and engineers will be able to appreciate why the main thrust of academic and industrial research in multimedia is concentrated on the matters covered in this chapter.

We will concentrate exclusively on TCP/IP networks. Bear in mind that this includes not only the Internet — although that was where these protocols were developed — but also 'intranets': local area networks utilizing the same set of protocols. Because of their generally higher speeds and more centralized management, intranets are better able to support some of the distributed multimedia applications, such as video conferencing, that can be implemented using the real-time protocols we will describe. For the Internet, universal deployment of these applications is still in the future.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 516
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Protocols

Protocols are the rules governing the exchange of data over networks. They are conceptually organized into layers, stacked on top of each other, with the lowest layer protocols dealing with the actual physical signals transmitted over wires and optical fibres. Above these, slightly higher level protocols handle the transfer of packets of raw data, and ensure that they reach their correct destination. The highest layers implement more application-oriented services, such as file transfer or Web browsing. The protocols on each layer are implemented using those on the layer below.

When a Web server and client exchange information using HTTP, for example, it appears as though the client sends HTTP requests to the server, which replies with HTTP responses. To the Web software, no other protocols appear to be involved, but in fact, what actually happens is that the HTTP messages are translated into streams of TCP packets, which are themselves translated into the format of the actual networks over which the data must pass; at the receiving end, these packets are used to reconstruct the incoming HTTP message. Thus, we need a little understanding of the lower level TCP/IP protocols before we can appreciate the characteristics of the application-oriented protocols that are used for multimedia.

TCP/IP networks are *packet-switched* networks, which means that all messages transmitted over the network are split into small pieces, called *packets*, which are sent separately. This enables network bandwidth to be shared efficiently between many messages, since, if the rate at which packets are generated for one message is lower than the available carrying capacity, packets belonging to other messages can be transmitted at the same time; we say messages are *multiplexed*. Contrast this with the telephone network, where a connection is established between the caller and the person being called, and this connection is held open for the exclusive use of their conversation until the caller hangs up, whether or not anybody is speaking. The advantages of this *circuit-switched* approach, which can be used for data exchange as well as spoken conversations, is that there is no need to distinguish between packets belonging to different messages on the same circuit — there is only ever one message — and that the two parties can rely on the availability of all the bandwidth provided by their circuit. On a packet-switched network, neither of these is the case: packets must carry additional

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 517
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

information identifying (at least) their source and destination, and the rate at which packets can be transmitted between two points will depend on what other traffic the network is carrying. For real-time distributed multimedia applications this last factor is particularly problematical.

Network and Transport Protocols

IP[1] is the *Internet Protocol*, both in the sense that that's what IP stands for, and in the more fundamental sense that it's what makes internets and the Internet possible. IP defines a basic unit of transfer, called a *datagram*, and provides a mechanism for getting datagrams from their source to their destination through a network of networks. The machines that actually exchange messages — the sources and destinations of datagrams — are called *hosts*. Each host will be connected to one of the networks making up the internet. It is identified by its *IP address*, a set of four numbers uniquely identifying the network and host, which is used by IP to send datagrams to the right place.

1 It's impossible to talk about networks without using sets of initials and acronyms — that's how network people talk. We've tried to keep them to a minimum, and will always tell you what they stand for.

⇨ IP addresses have a hierarchical structure; the assignment of numbers at the top level of the hierarchy is done by IANA. The organizations to which the top-level addresses are assigned take responsibility for assigning sub-addresses at the next level down, and so on. (ISPs sometimes assign IP addresses to their dial-up customers dynamically.) In this way, the uniqueness of IP addresses is ensured.

Datagrams belong to the category of data structures comprising a header, containing administrative information about the datagram, followed by the actual data. The header contains the source and destination IP addresses, as well as some additional information concerning routing and optionally security. The detailed layout of an IP datagram depends upon the version of IP being used. IPv4 is the 'classical' IP that has been in use for many years; IPv6 is a newer, more flexible, version. An important difference is that, whereas IPv4 addresses are 32 bits long, IPv6 addresses are 128 bits long, so the new version can accommodate many more hosts and networks. A datagram may be made up out of several packets on the underlying physical network that is carrying it.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 518
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

As well as hosts, an internet includes machines called *routers*, connected between its component networks, which maintain information about the network topology so that they can work out where to send a datagram to. In general, an IP datagram may have to pass through several routers during its travels over several networks. By inspecting the destination address in a datagram's header, a router can determine whether or not the datagram should be sent to a host connected to the same network as it is. If it should, the router translates the IP address to the native address format of the network, if necessary, and sends the data to its destination. Otherwise, the datagram is passed on to another router. Dynamically updated tables are used to determine which of the accessible routers a datagram should be forwarded to.

IP treats each datagram individually, recognizing no connection between the datagrams that make up a Web page, for example. Furthermore, it cannot identify the application that generated a piece of data, nor the one that should receive it. All that IP does is attempt to deliver individual datagrams from one host to another. It doesn't even guarantee to succeed. In particular, if some datagram has failed to reach its destination after a certain length of time,[2] it will be discarded. This is more efficient than continuing to try to deliver it — its destination is probably unreachable for some reason — and has the beneficial side-effect of preventing rogue datagrams going round and round forever. It follows, though, that if an application sends a stream of data, it may arrive at its destination with some pieces missing. It is also possible for some pieces to arrive in the wrong order. Since IP treats each datagram individually, and calculates routes dynamically, it is possible for a packet to overtake others sent before it. For example, a router that was down, forcing data to be sent round by a long route, may come back on line, and later packets will take a shorter, faster route to their destination, arriving earlier than packets sent before them.

2 In practice, after it has passed through a certain number of routers.

The majority of applications that communicate over TCP/IP networks require the data they send to arrive intact and in the right order at the receiving end. Consider an email system, for example. If we send you a message, you won't consider it very satisfactory if it arrives in your mailbox with key passages missing, and others in the wrong order. If applications had to rely on IP alone, this could happen, so, to provide the required service, the application would have to implement some mechanism for putting packets back in order, and requesting the retransmission of any that are

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 519
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

missing. It would be unreasonable to expect every application to implement these functions; instead, a reliable transport protocol, *TCP (Transmission Control Protocol)*, is layered on top of IP.

TCP provides reliable delivery of sequenced packets. It does this using a system of acknowledgement. A simplified version of the protocols would work as follows. When the destination receives a packet, it sends an acknowledgement, and the sender does not transmit another packet until it gets the acknowledgement. If the acknowledgement is not received within a specified period of time (the *time-out*), the packet is sent again. The algorithm as described is prohibitively slow, and TCP actually uses a more sophisticated technique, based on the same principle, using a 'sliding window' of unacknowledged packets. Instead of sending a single packet and waiting for an acknowledgement, it sends several, let's say eight for concreteness. As each packet is received, it is acknowledged; because of the time packets and acknowledgements take to traverse the network, the acknowledgement of the first packet will probably not reach the sender until several others have been sent. That is, there may be several unacknowledged packets in transit at one time. If the acknowledgement of the first packet has not been received by the time the eighth has been sent, the sender waits, and retransmits, as before, if the acknowledgement does not come in time. On the other hand, once the acknowledgement of the first packet is received, the limit on the number of packets to send before waiting is incremented, so the sender will send up to the ninth packet before waiting for the acknowledgement of the second, no matter at which point in the transmission of the first eight the acknowledgment arrives. Once the second acknowledgment is in, the limit is advanced to packet number ten, and so on (see Figure 15.1).

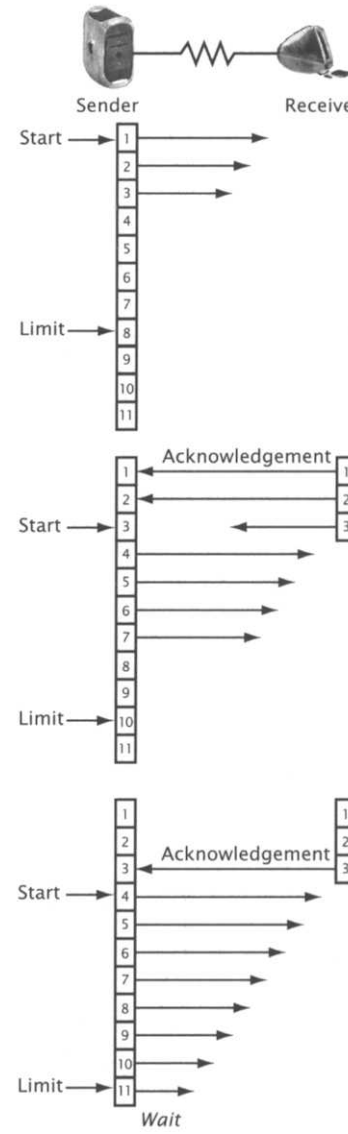


Figure 15.1
Sliding window acknowledgement

Notice that retransmitted packets may arrive out of sequence, and also that the possibility exists for packets to be sent twice, if an acknowledgment goes astray, for example, or a packet turns up long after it should. TCP thus introduces new opportunities for packets to arrive out of sequence. However, the finite number of packets that is allowed to be transmitted without acknowledgment imposes an upper bound on how far out of sequence a packet can be.

An analogy that is often used to describe the difference between IP and TCP concerns pouring water on to a burning building. IP is like a bucket brigade: water (data) is is poured out of a hydrant into buckets (datagrams) that are carried by a disorganized crowd

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 520
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

of volunteers (the network). A bucket may be carried some distance, then passed from hand to hand to the next volunteer. Some buckets will get dropped in the process; others will work their way through the crowd at different speeds, so that the order in which they arrive at their destination bears little resemblance to the order in which they are filled up and sent on their way. In contrast, TCP is like a firehose that carries the water straight from the hydrant to the fire; the water just goes in one end and comes out the other, in an orderly stream, without loss.

The analogy is not perfect. A firehose is a much more efficient way of transporting water than a crowd of bucket carriers, but TCP is less efficient than IP, because it is actually implemented using IP, but then adds extra overheads with its system of acknowledgments and retransmission.

Besides the overhead incurred by acknowledgement, TCP also incurs a cost in setting up a connection. This is necessary, in order for the acknowledgements to work, and in order to ensure that packets are sent to the appropriate application. Recall that IP addresses only identify hosts, and that IP accordingly only attempts to transport data between hosts. In order to identify a particular application running on a host, the IP address must be extended with a number, called a *port number*. Ports are associated with applications according to the protocol they use. For example, programs communicating via HTTP conventionally use port 80. The combination of an IP address and a port number is called a *transport address*, and enables the protocols running on top of IP to provide communication facilities between programs running on different hosts. TCP uses a pair of transport addresses to set up a connection between two programs, and subsequently these can communicate as if there was a reliable data conduit between them.

For some networked multimedia applications, the possibility of lost packets is more acceptable than the overhead of TCP. Streamed video and audio are the prime examples. Suppose, for example, a speech is being transmitted live over a network. If the video data were sent using TCP, then every frame would arrive complete and in the correct order, but the rate at which they would arrive would be unpredictable. Jerks and pauses might occur as packets were retransmitted and acknowledgements were awaited. Almost inevitably, the display would fall behind the actual delivery of the speech. The transient glitches caused by the occassional loss of a frame or fragment of audio would be less intrusive, if

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 521
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

they permitted an otherwise steady frame rate to be maintained instead. An alternative to TCP is needed to transport data with these characteristics.

The *User Datagram Protocol (UDP)* is built on top of IP, like TCP, but is much simpler, doing little more than ensure that datagrams are passed to the correct application when they arrive at their destination host. Like IP, UDP only tries its best to deliver datagrams, it does not offer the reliable delivery provided by TCP. Nor does it set up connections: port numbers are included in the source and destination fields of every UDP packet to ensure data finds its way to the right place. UDP does perform some elementary error checking that is not provided by IP, to help ensure that datagrams are not corrupted.[3] These features of UDP make it a suitable basis for building protocols for delivering data such as streamed video and audio, for which real-time constraints are more important than totally reliable delivery.

[3] We might say UDP notices if water sloshes out of the buckets, extending the analogy we used earlier.

The low-cost delivery of UDP is not enough, on its own, for such purposes. The *Real-Time Transport Protocol (RTP)* typically runs on top of UDP, adding extra features that are needed for synchronization, sequencing, and identifying different types of data — or *payloads* as they are called in this context. RTP itself still does not guarantee delivery and it does not prevent packets arriving in the wrong order, but it does enable applications to reconstruct a sequence of packets and detect whether any are missing. It does this by including a sequence number in the header of each packet belonging to a particular stream — RTP sets up connections between applications, so that the stream of data belonging to a particular connection is an identifiable entity. Each time a packet is sent over a connection, the sequence number is incremented; thus, it is clear to the receiver what order packets should be in. Depending on its requirements and the way in which it treats data once it has arrived, a receiving application can reconstruct the correct sequence, discard packets that are received out of sequence, or insert them in their correct places in some data structure being built from them.

An RTP packet's header also identifies the payload type — whether it is video, audio, and so on — which determines, by implication, the format of the data contained in the rest of the packet. This allows different payload types to employ formats that are optimized for their special characteristics; in particular, it allows appropriate forms of compression to be applied to images, video, and sound. Where several different media types are being transmitted —

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 522
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

typically video with accompanying sound — they must be sent over separate RTP streams with different payload types. It is therefore necessary to synchronize them when they are received. A timestamp is included in the header for this purpose. It records the instant at which the first byte contained in a packet was sampled. This can be collated with the timestamps in other, related, streams, to ensure that simultaneously sampled data is played back at the same instant.

RTP can be tailored to different applications' requirements and extended to cope with new types of payload — it is a framework for an entire family of protocols, rather than a protocol in the traditional sense. RTP *profiles* define subsets of the protocol's features that are suitable for particular applications. In particular, the audio-visual profile is intended for use with audio and video streams.

A complementary protocol, the *RTP Control Protocol (RTCP)* can be used in conjunction with RTP to provide feedback on the quality of the data delivery — statistics such as the number of packets lost and the variance of the time between packets' arrival. This data can be used by the application sending the data, to adjust the rate at which it does so, for example.

Multicasting

Previously, when we considered bandwidth limitations and how they affect online delivery of multimedia, we concentrated on the speed of users' connections to the Internet, via modems or cable and so on. These connections are usually the slowest link in the communication chain, but the high-speed networks that provide the skeleton of the Internet are also of finite bandwidth. Here, too, new technologies are leading to increased speeds, but traffic is also growing at a considerable rate, and new types of traffic — especially that associated with bulky time-based media such as video — are adding to the demands placed on the Internet as a whole. Speeding up the network is one way of coping with these demands; reducing the volume of traffic by eliminating duplication is another.

A common situation where data is unnecessarily duplicated arises when a group of Internet users require access to the same resource at the same time. This situation is often associated with multimedia.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 523
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Suppose, for example, a well-known musician — let's call him Tim Linkinwater — has arranged to transmit a live video feed from a concert over the Internet. Every Tim Linkinwater fan with Internet access will want to watch this Webcast. The data being sent to every fan is identical, but conventional (*unicast*) transmission will require that the server from which the video is being streamed send a copy of it to everybody who has set up a connection to watch this concert. If the concert is taking place in New York, but Tim Linkinwater is big in Europe, many copies of the video data will be sent across the Atlantic, putting a strain on both the transatlantic links and the video server. This strain could be reduced by sending a single copy over the Atlantic and not duplicating it until it became necessary to do so in order to distribute the data to all the people who wanted it.

The scenario just outlined is the basis of *multicasting*. Figures 15.2 and 15.3 illustrate the difference between unicasting and multicasting. In unicasting, a separate packet is sent to each user; in multicasting, a single packet is sent, and is duplicated along the way whenever routes to different users diverge. For this to be possible, hosts must be assigned to *host groups*, with certain network addresses identifying groups instead of single hosts. A range of IP addresses is reserved for this purpose. From the sender's point of view a multicast address behaves rather like a mailing list — although mailing lists are implemented quite differently (using unicast). One difference is that the sender does not know who belongs to the host group, if anybody.

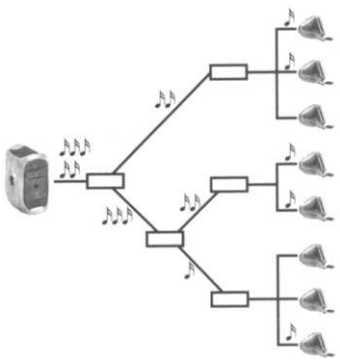


Figure 15.2
Unicast transmission

A packet that is intended to go to a group is sent to the appropriate multicast address. Routers must be enhanced to cope with multicast addresses, duplicating packets where necessary. They must also be able to perform the routing calculations necessary to determine optimal routes — which may be different from optimal unicast routes. These technical issues, while difficult, are amenable to technical solutions. Other problems that arise when multicasting is attempted on the scale of the Internet are less straightforward.

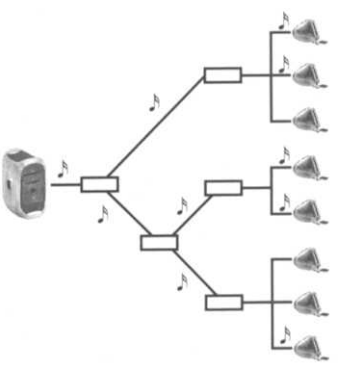


Figure 15.3
Multicast transmission

⇨ On a LAN, there are no routers, so multicasting is simpler. Multicast packets will go to the network interface on every host. Each host just has to look at the multicast address and determine whether it belongs to the corresponding host group; if it does, it accepts the packet, in the same way as it accepts a packet sent to its individual address. Multicast applications are therefore more common on LANs than the Internet, at present.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 524
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

The main problems concern management of host groups. These will typically be spread across different networks, so the information about which hosts belong to a group must be distributed to routers. Group membership has to be dynamic, to accommodate applications such as the Webcast described earlier, so a mechanism is required for a host to join a group, and subsequently to leave it. Routers must then be able to adjust to the new state of group membership. At a still higher level, some mechanism is needed for users to find out about the existence of host groups, what their multicast addresses are, and how to join them. (Again, compare this with discovering mailing lists.) Essentially, when a multicast session is set up, an address must be chosen from the available range, and then advertised — by email, newsgroup, Web site, or a dedicated application that serves the function of a listings magazine.

Many distributed multimedia applications, including live video streaming, and conferencing, have characteristics that make them suitable for multicasting. Protocols such as RTP can run on top of IP multicast.[4] Considerable experience of doing so has been gained using an experimental sub-network of the Internet, known as the *MBone (Multicast Backbone)*, and multicast capabilities are now being provided in routers as a matter of course. ISPs are beginning to offer multicast services, but this emergence of multicast into the world of the commercial Internet raises new questions. Above all, how should multicasting be charged for?

4 TCP cannot, however.

⇨ There is a third alternative to both unicast and multicast transmission: *broadcast*. A broadcast transmission is sent to every host on the network, and it is up to the host to decide whether or not to ignore it. This is very efficient for the sender, and it may be an efficient way of using network bandwidth, but for most purposes it imposes an unacceptable load on hosts, since it requires them to look at every packet sent to a broadcast address and determine whether it is interesting. Broadcasting is used on the Internet for monitoring the status of the network and for maintaining routing information, amongst other things.

Application Protocols for Multimedia

The network and transport protocols we have described do no more than deliver packets of data, more or less reliably, to their

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 525
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

designated destinations. Higher level protocols must run on top of them in order to provide services suitable for distributed multimedia applications. We will describe two such protocols: HTTP, which is the basis of the World Wide Web, and RTSP, a newer protocol designed to control streamed media. Our description is not exhaustive, but is intended to show what is involved in mapping the requirements of some kinds of distributed multimedia applications on to the transport facilities provided by the protocols described in preceding sections.

HTTP

Interaction between a Web client and server over HTTP takes the form of a disciplined conversation, with the client sending requests, which are met by responses from the server. The conversation is begun by the client — it is a basic property of the client/server model that servers do nothing but listen for requests, unless they are asked to. To start things off, the client opens a TCP connection to a server. The identity of the server is usually extracted from a URL. As we saw in Chapter 9, the URL contains a domain name; TCP only works with numeric IP addresses, though, so the name must be translated into an address. This translation is done transparently, as far as HTTP is concerned, by DNS (Domain Naming Services), so HTTP can always work in terms of names (although numeric addresses are occasionally used instead). By default, Web servers listen to port 80, so this will normally be the port number used when the connection is opened.

➲ Originally, prior to version 1.1 of HTTP, a connection was used for each request and its response only: that is, the client opened a TCP connection and sent one request; the server sent a response to that request and closed the connection. This way of using connections is very efficient from the server's point of view: as soon as it has sent the response and closed the connection it can forget about the transaction and get on with something else, without having to keep the connection open and wait for further requests to come over it, or close the connection if it times out. The disadvantage is that accesses to Web servers tend to come in clusters. If a page contains ten images most browsers will make eleven requests in rapid succession — one for the page itself and one for each image — so eleven connections between the same pair of hosts must be opened and closed. For technical reasons, to help minimize lost packets, a TCP/IP connection starts out at a slow rate and gradually

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 526
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

works up to the best speed it can attain with acceptable losses in the existing state of the network. Hence, in our example, the data for the page and its images will not be sent as fast over the eleven connections as it would over a single connection, since each new connection has to start over and work up to speed. HTTP version 1.1 has therefore introduced the possibility of a persistent connection, which must be explicitly closed by the client (although the server will close it after a specified time has elapsed without requests). Nevertheless, the structure of an HTTP session retains the form of a sequence of requests that evoke responses. No state information is retained in the server, which is therefore unaware of any logical connections between any requests.

HTTP requests and responses are collectively known as messages. Both consist of a string of 8-bit characters, so they can be treated as text by programs that read them (and can be read by humans if they have a program that can eavesdrop on HTTP). Messages conform to a simple rigid structure, consisting of an initial line (the *request line* for a request, the *status line* for a response) containing the essential message, followed by one or more *headers*, containing various parameters and modifiers. These may be followed by the message *body*, which contains data, such as the contents of a file being sent by the server, if there is any. Headers are separated from the data by a blank line.[5]

5
On the Internet, the combination of a carriage return followed by a linefeed is defined as the line terminator; this is not the same convention as all systems use, so HTTP clients and servers must take care with line termination. Not all do so.

A request line comprises three elements. The *method* is a name identifying the service being requested; the most commonly used method is GET, which is used to request a file or other resource (we will not consider other methods in any detail). The *identifier* comes next, and tells the server which resource is being requested, for example by giving the path name of a file. Finally, the HTTP *version* indicates which protocol version the client is using. For example, if a user clicked on a link with URL `http://www.wiley.com/compbooks/chapman/index.html`, their Web browser would connect to the host with name `www.wiley.com`, using port 80 so that communication would go to the Web server on that machine. It would then send an HTTP request, whose request line was:

```
GET /compbooks/chapman/index.html HTTP/1.1
```

The headers that may follow a request line all take the form of a header name followed by a colon and some arguments. For example, the following two headers can be found after the request line shown above:

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 527
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

```
Host: www.wiley.com
User-Agent: Mozilla/4.0 (compatible; MSIE 4.5; Mac_PowerPC)
```

The `Host` header tells the server the host name that the request is directed at;[6] `User-Agent` identifies the Web browser (or other user agent) that is making the request. This will allow the server to make allowances for any known problems with this particular browser, if it chooses to.

[6] Sometimes, several host names correspond to a single IP address, and this might make request identifiers ambiguous.

One of the most commonly seen headers in `GET` requests is `Accept`. Its arguments indicate, using MIME content types, the range of types of data that the browser can deal with. For example,

```
Accept: image/gif, image/x-xbitmap, image/jpeg
```

is sent by a browser to indicate that it is capable of and willing to display GIF, JPEG and X bitmap images. Browsers may send several `Accept` headers instead of combining them. Web servers will only send them data that is of an acceptable type. In a MIME type used in this way, a * can be used as a wildcard character, so

```
Accept: text/*
```

indicates that the browser will accept any text format.

The value specified in an `Accept` header can be modified by two qualifiers. The first is a 'quality factor', `q`, which expresses the desirability of a type, using a real number between zero and one. Attaching different quality factors to the sub-types of the same MIME type allows a Web client to express a preference among different formats. For example,

```
Accept: image/png;q=0.8
Accept: image/jpeg;q=0.6
```

indicates that PNG files are preferred over JPEGs. The second modifer, `mxb`, signifies the maximum number of bytes that will be accepted in files of the designated type. If a browser only preferred PNG files over JPEGs if they were not too big, the following headers could be used:

```
Accept: image/png;q=0.8;mxb=30000
Accept: image/jpeg;q=0.6
```

As the examples show, the modifiers are separated from the type and from each other by semi-colons.

Despite the potential usefulness of this header in allowing servers to offer alternative versions of images and other content, and negotiate with clients to choose the most appropriate, most browsers appear to send

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 528
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

```
Accept: */*
```

among their headers by default.

Two similar headers, `Accept-Charset` and `Accept-Language` are used by browsers to inform servers about the character sets and languages they will accept. The following would be typical of the headers sent by a browser set up in an English-speaking country:

```
Accept-Charset: iso-8859-1
Accept-Language: en
```

Since a `GET` request does not send any data, its body is empty — the message terminates with the blank line.

The first line of an HTTP server's response is the status line, indicating how it coped with the request. This line begins with the protocol version, telling the client which HTTP version the server is using. Next comes a numerical status code, whose meaning is defined by the HTTP standard, followed by a short phrase, explaining to human readers what the code means. If all goes well, the code will be 200, which means `OK`, as shown:

```
HTTP/1.1 200 OK
```

Just as the client told the server who it was, in the `User-Agent` header, the server introduces itself in the `Server` header. It also dates and times the response, and, as we saw in Chapter 9, uses the `Content-type` header to inform the client about the MIME type of the data being returned. For example:

```
Server: Netscape-Enterprise/3.5.1G
Date: Sat, 17 Jul 1999 14:27:17 GMT
Content-type: text/html
```

We will describe some other noteworthy headers shortly.

Typically, a server's response does contain some data; in the case of a response to a `GET` request, this will be the contents of the file that was requested, for example.

> Where a persistent connection is being used, the client must be able to determine when the end of the data has been reached — when servers closed the connection immediately after sending the response this was not a problem, because data obviously finished when the connection closed. Two mechanisms are now provided. Either a `Content-length` header can be sent, or the response can be broken into 'chunks', each of which is self-delimiting. For the details, consult the HTTP 1.1 standard.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 529
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

The status code in a response is not always 200, unfortunately. The HTTP 1.1 standard defines many 3-digit return codes. These are divided into groups, by their first digit. Codes less than 200 are informative; the only such codes presently defined are 100, which means 'continuing', and is only applicable to persistent connections (see the technical asides earlier in this section), and 101, sent when the server switches to a different protocol, for example to stream some real-time data. Codes in the range 200–299 indicate success of one sort or another. Those in the three hundreds are used when the server must redirect the request to a different URL. For example, a code of 301 ('moved permanently') indicates that the resource identified by the URL in a `GET` request has been permanently moved to a new location (the server sends a `Location` header giving the new URL).

Codes that begin with a 4 or a 5 represent errors, by the client and server, respectively. Probably the two most commonly encountered error codes are 400 and 404. 400 means 'bad request' — the request was malformed in some way. This error usually occurs when requests are manufactured by programs other than Web browsers. Error code 404 is 'not found' — the URL in a `GET` request does not correspond to any resource on the server. This is the code that results from broken links. Other error codes in this range correspond to requests for services that are denied by the server, such as attempts to access protected files. Another interesting code is 406 ('not acceptable'), which is sent if the requested resource's MIME type is not listed in the `Accept` headers of the request. Server errors include 500 ('internal server error'), which should never happen, and 501 ('not implemented'), which is sent when the server does not know how to implement a requested method.

⇨ Where confidential information is being transmitted between a Web client and server — credit card details being supplied to an online store being the most prominent example — the transactions must be secure. Mechanisms have been added to HTTP for guaranteeing that a server really is what it claims to be, and for encrypting all the data sent between the server and client.

Caching

The World Wide Web has made information accessible over the Internet in a way that it had never been previously. This has led

to an increase in network traffic as people take advantage of this new-found accessibility. In order to help alleviate the strain this could cause on the network, HTTP provides support for a system of *caching*, which allows copies of pages that have been received to be kept on a user's machine, and on other intermediate machines between it and the server from which they originate. When a page is requested more than once, it can be retrieved after the first time from a cache instead of from the server. If the cache is on the local machine, this may not require any network access or action by the server at all.

The full extent of HTTP 1.1's support for caching is quite extensive, supplying several alternative mechanisms; we will only describe a subset.

The trouble with caching is that a version of the requested resource that is newer than the version in the cache might have appeared on the server. In that case, the new version should be retrieved, and the old one in the cache should be discarded. The simplest way for the client to find out whether that is the case is to ask. It does so by sending an `If-Modified-Since` header, giving the date and time of its cached copy (which it knows from the `Date` header in the response to its original request). The request is then said to be *conditional*, and the server only sends the requested page if the condition is satisfied, that is, if the page has been modified since the date specified in the header. If it has, the server sends a response containing the modified page, as before. If not, it sends a response with status code 304, which means 'not modified'. On receiving this code, the browser displays the page from its cache.

Conditional requests of this sort eliminate the need for servers to send complete responses in all cases, thus reducing the volume of data transmitted. A further facility potentially eliminates the need for servers to send any response, by eliminating the need for clients to send requests. A server can include an `Expires` header in its response to a `GET` request, indicating the date and time after which the data it returns should no longer be considered up-to-date. Until that time, the client is free to use a cached copy of the data to satisfy any subsequent requests for the same resource. Thus no network activity at all is required to obtain the same page until it reaches its expiry date.

While this mechanism is sound, it begs the question of how servers are to assign expiry dates to arbitrary pages. Most of them appear not to try, although one can imagine tightly controlled

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 531
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

environments in which it would be known how long a page would remain valid.[7] In the absence of any widespread use of `Expires` headers, Web clients fall back on their own *ad hoc* devices for avoiding unnecessary network accesses. For example, Internet Explorer offers an offline browsing mode, in which all requests are met from the local cache, if possible. This allows previously visited Web sites to be accessed without the user even being connected to the Internet.

7 The HTTP standard suggests the use of heuristics based on, for example, the `Last-modified` header, but does not specify any detailed algorithms. It recommends that heuristics be applied with caution.

So far, we have assumed that a cache is maintained on a user's machine. There are other places that data can be cached, though, sometimes more effectively. In particular, *Web proxies* usually maintain large caches. A proxy is a machine that handles requests directed to some other server. So when a client that is configured to use a proxy sends a request, the request is sent to the proxy, which then forwards it to its designated destination, receives the response and passes it back to the client. This apparently pointless exercise is often needed when clients are operating behind a 'firewall' — a specially modified router that filters packets to provide additional security. Firewalls prevent users inside them from making HTTP requests, so a proxy is used, which has access to the outside world and the protected machines. Security can be added to the proxy to prevent unauthorized access across the firewall, in either direction. This implies that all responses to requests sent by any machine inside the firewall pass through the proxy, which means that a cache on the proxy will end up holding a pool of data that is of interest to people in the organization maintaining the firewall. This is likely to mean that many requests can be met from the proxy's cache.

Proxies can provide effective caches whenever they are placed somewhere that many requests pass through. Large ISPs, in particular, can employ proxies with extremely large caches to intercept a large proportion of their customers' requests.

Caches cannot help with pages that are dynamically generated, as an increasing number are, in ways that we will describe briefly in a later section. Nor do they help users who never visit the same page twice unless it has been updated. And caches are finite, so that, sooner or later, cached copies of pages must be discarded, and retrieved again if they are requested subsequently. Nevertheless, they are widely considered to be making a useful contribution to keeping network traffic within bounds.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 532
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

RTSP

HTTP provides the services necessary for implementing a distributed hypertext system: it should be easy to see how HTTP requests can be constructed in response to mouse-clicks on links, and how a browser displaying the data in their responses can provide the experience of following a link to a new page. HTTP can cope with embedded images, sounds, and video, by downloading a complete data file. Streaming audio and video require a different treatment, though.

In the first place, HTTP runs on top of TCP, and, as we explained earlier, the overheads incurred by the reliable delivery that TCP provides are unacceptable for streamed media, which are better served by a less reliable but more efficient protocol. Its use of TCP also makes HTTP unsuitable for multicasting. RTP, as described earlier, is adequate for carrying streaming media data, and can be used for multicasting, but it does not provide all the necessary functionality required by streams of such data. We usually want to be able to start, stop, and pause them, and possibly (for streams that are not being transmitted live) go to a particular point in the stream, identified by a timecode. For live streams, we might want to schedule a time at which to start the display; for example, if a concert was being transmitted over the Web, we might want to skip the support band and start listening when the headline act was due to go on stage.

The *Real Time Streaming Protocol (RTSP)* is intended to provide these services. It is often described as an 'Internet VCR remote control protocol', which conveys much of the flavour of what it does. Syntactically, it closely resembles HTTP, with request and status lines and headers, many of which are the same as in HTTP, but there are some differences. In particular, RTSP uses ISO 10646 (with UTF-8 encoding) as its character set, unlike HTTP which is restricted to ASCII in its messages.[8] In RTSP requests that include an identifier, an absolute URL must be used, instead of the pathname that may be used in HTTP; RTSP consequently does not need a separate `Host` header. Most importantly, RTSP responses do not carry the media data, the way that HTTP responses carry Web page data. Instead, the data is transmitted separately, often using RTP; RTSP merely coordinates the transmission.

8 But not its data, of course.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 533
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Before an RTSP session can be set up, the client must obtain a *presentation description*, which contains information about the data streams that are to be controlled, which together make up a multimedia presentation. The RTSP specification does not stipulate the format of a presentation description; the *Session Description Protocol (SDP)*'s format is commonly used, but the prevailing applications using RTSP (Streaming QuickTime and RealPlayer G2) take advantage of that format's extensibility to customize it to their own requirements, so, in effect, presentation descriptions largely conform to proprietary formats.[9] A typical description will include one or more *media announcements*, including the transport address and protocol (e.g. RTP) of one of the session's component streams, and information about the encoding used, including the type of compression, if any. Each stream will have an `rtsp://` URL. It is quite possible for different streams to be served from different hosts. The presentation description will also provide a *connection address*, to which subsequent requests are addressed. This is necessary because RTSP does not specify the mechanism whereby presentation descriptions are to be retrieved. There is a `DESCRIBE` request that is often used for this purpose, but it is permissible for the presentation description to be retrieved using HTTP, or even sent to a user by email.

9 These are fairly readily comprehensible, though.

In practice, both RealPlayer and Streaming QuickTime use `DESCRIBE` requests, but need a bootstrapping step before they can find out where to send it to. Typically, HTTP is used to obtain the URL of the presentation description; this will begin with `rtsp://`, to indicate that it can be accessed using RTSP. QuickTime uses a self-contained approach: a small movie containing URLs for different versions of a session corresponding to different connection speeds is presented as the representation of a streamed movie[10] — for example, it might be embedded in a Web page, where a user can click on it, causing it to be retrieved by HTTP. RealPlayer uses a small file with a distinctive extension for the same purpose, although it does not provide for different versions.

10 This is just a special case of the use of a master movie to orchestrate a set of alternatives, as described in Chapter 10.

Once the necessary URLs have been obtained by the client, it sends a `DESCRIBE` request, to which the server responds with a message containing the required presentation description. The client's request can include an `ACCEPT` header, indicating the description formats it understands. Once it has a presentation description, the client can send a `SETUP` request. The server responds with a message that includes a *session identifier*, an arbitrary string that

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 534
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

is used by both client and server to identify messages associated with the same session. (RTSP does not use connections, the way TCP does, partly because the actual streamed data is sent over a separate transport protocol, and partly to enable it to be used for controlling multicast streams.) Everything is then in place for the streaming to begin.

⇨ A significant difference between the way QuickTime and RealPlayer G2 use RTSP lies in their approach to presentations comprising several streams, such as a video stream and accompanying audio. QuickTime treats these as separate tracks of a single movie (although they will be transmitted separately), so the presentation description provides details of every stream. RealPlayer G2 is a SMIL player (see Chapter 13), and uses SMIL to coordinate the separate streams. The presentation description it originally retrieves only describes a single SMIL document; this contains `rtsp://` URLs for each of the streams in the presentation, and the player sets up separate RTSP sessions for each.

In the simplest mode of operation, an RTSP client sends **PLAY** requests to cause the server to begin sending data, and **PAUSE** requests to temporarily halt it. As we mentioned earlier, a **PLAY** request may include a header specifying a range within the duration of the stream; SMPTE timecodes or clock values in a standard format may be used for this purpose. A session is ended by the client sending a **TEARDOWN** request, which causes the server to deallocate any resources associated with the session. It is, of course, conceivable that something might go wrong. Status codes are used in responses to indicate any errors that might occur, just as they are in HTTP. Indeed, wherever it makes sense, RTSP uses the same status codes as HTTP.

We emphasise that RTSP messages travel separately from the streamed data they relate to. Usually, RTSP will operate on top of TCP, so that it can rely on delivery of its messages, although it can as easily run on top of UDP, where efficiency is important. For the streamed data, there is no such choice. As we explained previously, TCP is not suitable for carrying the streamed data itself, and this will usually be carried over RTP, which in turn runs on top of UDP. Nevertheless, there is always a logical connection between the streamed data and the RTSP messages. For example, RTSP supports the same headers for controlling caching as HTTP does, but, although the headers are transmitted as part of RTSP messages, it is the streamed data that is cached, not the messages.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 535
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

RTSP can be used for unicast and multicast streams. Where unicast is used, it provides services such as video on demand: a client requests, say, a movie to be streamed from a server, and the data is sent exclusively to the client who controls its playback with RTSP requests. Multicasting provides services more like conventional television broadcast: a client joins a multicast session that has been set up beforehand, to transmit a concert, for example. Where RTSP is being used with a multicast service, the main difference from the client's point of view is that the server has to tell it the multicast address for the transmission, whereas, in the unicast case, it is the client that has to tell the server where to direct the data stream to.

Figure 15.4 summarizes the relationships between the protocols we have described. Video conferencing and similar applications require additional protocols for session management and the equivalent of rules of procedure. We will not describe the protocols presently available for these purposes, since they are still largely experimental, and take us too far away from our main concerns.

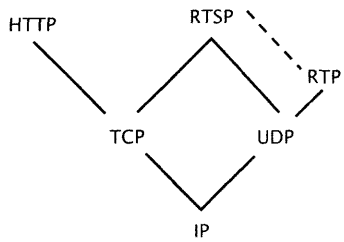


Figure 15.4
Relationships between TCP/IP protocols used for multimedia

Quality of Service

Normally, when data is being transmitted over a packet-switched network, it is subject to the vagaries of network performance, which depends on many factors, most of them unpredictable. This can interfere with the satisfactory streaming of media in several ways. First, *delays* can be introduced. There is always some delay due to the distance signals must travel at a finite speed; the operations that must be performed by routers contribute additional delays. Whereas traditional network applications, such as email and file transfer can tolerate considerable delays, real-time and streamed multimedia are much less tolerant. You are probably familiar with the slightly disconcerting effect of the transmission delays that occur when live interviews are conducted over transatlantic links for television news bulletins. The effect of similar, or longer, delays on a real-time application, such as a video conference, can seriously interfere with its effectiveness. Streamed applications can tolerate longer delays, although, the appeal of live streaming often lies in its immediacy, which is compromised by excessive delays.

Typically, the delay experienced by a particular stream of data will not be constant, but will vary continuously. The variation in delay is

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 536
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

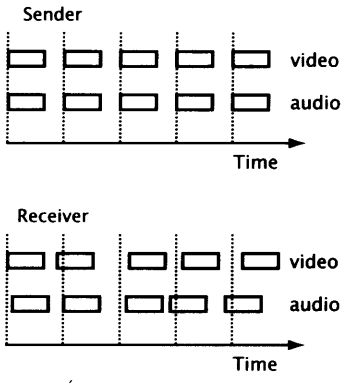


Figure 15.5
Effect of delay jitter on synchronization

called *jitter*; it means that the time between the arrival of successive packets will vary. Jitter causes two kinds of problem for multimedia. First, the variation in the time between packets can result in time base errors; that is, samples will be played back at the wrong time. This is most likely to affect audio streams, where it manifests itself as noise. Second, where a presentation comprises several streams that are transmitted independently, as they will be over RTP, for example, the streams may jitter independently, resulting in a loss of synchronization between them (see Figure 15.5). Unless this can be corrected at the receiving end, disturbing — or unintentionally comical — effects may occur, since people are sensitive to tiny errors in lip-sync. As with delay, different applications can tolerate different amounts of jitter. For example, a video application might buffer incoming data streams, so that it could smooth out jitter and synchronize video and audio streams, provided the jitter was within certain limits.

In addition to delay and jitter, there may be packet loss, since, as we explained earlier, streamed media cannot be sensibly transported using a reliable protocol such as TCP. Once again, different applications can tolerate different amounts of packet loss.

Delay, jitter, and packet loss are measurable quantities. We can, therefore, quantify the amounts of each which a particular application can tolerate. The values of these parameters, together with the bandwidth it needs, define the *quality of service (QoS)* required by an application.

Established types of network service do not require much in the way of quality of service. Mostly, they are based on the transfer of entire files, such as a mail message or the HTML and embedded images of a Web page. They can tolerate as much delay as users' patience allows, and jitter is irrelevant, since nothing will be done with the data until it has all arrived. However, it is generally the case that packet loss is completely unacceptable. These characteristics led to the development of transport protocols like TCP that are reliable but pay no other attention to the requirements of applications. On the contrary, their performance is driven by the availability of bandwidth, and their objective is to make the best use of that bandwidth — which is how delay and jitter are introduced, since the bandwidth available to any one connection is not constant or guaranteed. Integrating support for streamed and real-time applications with their more demanding requirements for quality of service is thus quite a challenge.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 537
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

New high-speed network technologies are coming into use, which are able to reserve resources for individual connections. The best known of these is *ATM*, which stands for *Asynchronous Transfer Mode.* A pure ATM network can offer guarantees about the quality of service it will provide. Applications can reserve resources to satisfy their requirements, and the network will honour those reservations, so that the application can be assured that delay, jitter and packet loss will be kept within tolerable limits. In order to maintain the quality of service to some applications, the network may have to restrict the resources it allocates to others, perhaps even refusing to accept any new traffic after a certain point. The network is thus behaving much more like a circuit-switched network, such as the telephone system, than an ordinary packet-switched network.

Making an open heterogeneous network of networks, such as the Internet, behave in the same way, while still providing the traditional services that users have come to expect, is problematical. Parts of the Internet are presently ATM networks, but other parts are not, so the quality of service support has to be implemented at a higher level, by way of protocols. A framework known as the *Integrated Services Architecture* has been developed for this purpose.[11] It defines different classes of service, such as 'best effort', which is the traditional IP approach to delivery, and 'guaranteed', where an absolute upper bound is imposed on the delay of packets. Several mechanisms for controlling network traffic to ensure that specified classes of service are actually provided are defined, together with a protocol for reserving resources, *RSVP (Resource Reservation Protocol.* Some effort has been made to ensure that the facilities of the Integrated Services Architecture work with multicasting, since it is expected that many applications, such as video conferencing and live video streaming, requiring guaranteed quality of service will employ multicasting.

11 At the time of writing, it is only experimental.

⇨ One thing seems fairly certain: when ISPs start providing guarantees about quality of service, they will make additional charges for them. It is quite likely, therefore, that streamed video with guaranteed performance will not be something that can be provided for everybody. More likely, when (or if) the Internet can provide such services, it will enable established broadcasting companies to use the Internet as an alternative delivery medium for certain sorts of content. This is already being seen, with organizations such as CNN, Disney, FoxTV and the BBC providing Real Video and Streaming QuickTime channels, albeit subject to the vagaries of ordinary IP delivery. Very probably, when these services evolve to

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 538
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

provide higher quality interactive TV, they will only be provided on a subscription basis. The notion that streaming video technology can turn every Web page into a TV station may need some qualification.

Server-side Computation

In previous sections we described protocols that allow multimedia data to be carried over a network, but we have not considered where that data ultimately comes from. Consider the World Wide Web, in particular. Using client-side scripting, as described in Chapter 14, we can produce Web pages that are responsive to user input in a limited way, but they must be made up from a fixed collection of elements — typically stored in files accessible to a server, although video and audio streams can also be incorporated by a server that understands the appropriate protocols. Using this approach, we cannot even embed the time of day in a Web page.

Server-side scripting is used to enable an HTTP server to communicate with other resources, such as databases, and incorporate data obtained from them into its responses. In particular, server side scripts are used to enable a server to construct Web pages dynamically from time-varying data. The *Common Gateway Interface (CGI)* provides a standard mechanism for communication between a server and server-side scripts, although, as we mentioned in Chapter 2, proprietary scripting interfaces tied to a particular server are also widely used. Server-side scripting is not used much for multimedia yet, but as the limitations of client-side scripting become more apparent, and multimedia databases become more widely available, server-side scripting will become more prominent.

The Common Gateway Interface

Earlier in this chapter, we described how a client sends HTTP requests to a server, which sends responses back. The Common Gateway Interface provides a mechanism for the server to pass on data in a client's request to a script[12] — referred to as a *CGI script* — running on the server's machine, and for the script to pass data back, which the server then sends on to the client. An important point to realize is that the communication between the client and

12
Or program, but most server-side computation is done using scripting languages, especially Perl.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 539
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

server takes place using HTTP, so the output of a CGI script must be something from which the server can construct an HTTP response. Some scripts[13] actually generate complete HTTP responses, but more often a CGI script produces some headers, together with the response body, and leaves the server to add the status line and other headers, such as `Server`, which it is better placed to generate.

13 Technically known as 'non-parsed header' scripts.

CGI is a mechanism for communication in both directions between client and CGI script, via the server. A CGI script is called by the server when the identifier in the request line denotes a script (usually because it refers to an executable file in a designated script directory). How does the client send data to the script? There are three possible places.[14] First, there are the headers in the HTTP request. Some of these might contain information that is useful to a CGI script. Second, as we mentioned in Chapter 9, a *query string* may be appended to the end of the URL — it is then added to the identifier in the request line. Third, a similar string can be sent as the body of a request whose method is `POST`. We must begin by looking at the format of these query strings, and how they are constructed by browsers.

14 Actually, there's a fourth, but it is something of a security risk, and therefore is not used much any more, so we won't tell you about it.

A query string assigns values to named parameters. In this respect, it is rather like the `<PARAM>` tag in HTML, but its syntax is quite different. A query string consists of a sequence of assignments of the form *name* = *value*, separated by & characters. Within the *value*, spaces are encoded as + signs, and other special characters are replaced by code sequences in the same way as they are in URLs (see Chapter 9). Thus, an HTTP request for a CGI script called `register.cgi`, with a query string setting the value of a parameter called `Name` to the string `MacBean of Acharacle`, and that of `Profession` to `Piper` would look like this:

```
GET /cgi-bin/register.cgi?Name=MacBean+of+Acharacle&Profession=Piper HTTP/1.1
```

Such a request can be generated by any program that has opened an HTTP connection, but most commonly HTTP requests are constructed and sent by Web browsers. There isn't much point in using URLs with fixed query strings in anchors; to be useful, query strings need to be constructed dynamically from user input. The traditional way of doing this is by using HTML's facilities for constructing and submitting forms, comprising text boxes, radio buttons, pop-up menus, and check boxes. These facilities are quite elaborate, and we will not describe them in any detail. Suffice it to say that each element within the form has a `name` attribute, whose

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 540
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

value is used as a parameter name when the browser constructs a query string. The value of the parameter is taken from the user's input in the form element. Text fields are HTML `INPUT` elements with a `type` attribute of `text`, so a simple form that was used to elicit a user's name and profession for registration purposes might be constructed in the following way:

```
<FORM method="get" action="cgi-bin/register.cgi">
  <P>Your name:
    <INPUT type="text" name="Name" size="54">
  </P>
  <P>Your profession:
    <INPUT type="text" name="Profession" size="49">
  </P>
  <P>
    <INPUT type="submit" name="Submit" value="Submit">
  </P>
</FORM>
```

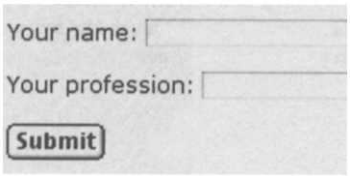


Figure 15.6
A simple HTML form

The displayed page produced by this code is shown (truncated) in Figure 15.6. The `method` attribute of the `FORM` element is used to determine the method in the HTTP request (`GET` or `POST`), and its `action` attribute identifies the CGI script. The `INPUT` element whose `type` is `submit` is a button that can be clicked on to submit the form. This causes the browser to send the HTTP request shown earlier to the server, which in turn passes the parameter values and request headers to the script `register.cgi`.

The mechanisms whereby the parameter values and headers are passed to the CGI script are of interest only to programmers. In fact, a muddle of different mechanisms is used, which can make writing CGI scripts from scratch excessively tedious. Fortunately, libraries exist which encapsulate all the tedious processing and provide a clean interface that allows the CGI programmer to access the values that the server has passed to the script with a minimum of effort. If you are going to write CGI scripts in Perl, Lincoln Stein's CGI module will save you a lot of effort. A brief introduction to its use can be found in Chapter 10 of *Perl: The Programmer's Companion* by Nigel Chapman (John Wiley & Sons, 1997). A full account is given in Lincoln Stein's own book: *Lincoln Stein's Guide to CGI.pm* (John Wiley & Sons, 1998).

As we remarked earlier, a CGI script must pass back to the HTTP server (most of) an HTTP response. Often, CGI scripts construct HTML by carrying out some computation based on the input they

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 541
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

receive, and place it in the body of their response; that is, they provide a means of building Web pages dynamically. They can, however, send data of any type in the response, providing they supply a suitable `Content-type` header specifying the appropriate MIME type. Browsers cannot distinguish between a response generated by a CGI script and one sent directly by an HTTP server,[15] and will display any type of data that is generated by a script, using plug-ins and helper applications where necessary. CGI can therefore form the basis of a distributed multimedia application, with a Web browser providing a front-end to elicit user input with a form and display multimedia elements generated by a script's computation. Often this computation will include accessing a database, possibly containing the multimedia data itself, or possibly just providing information about the data, which is stored separately. In that case, users will be able to construct queries specifying desired properties of the media they wish to retrieve.

15 This isn't quite true. In fact, they need to be able to distinguish between them, since it is not sensible to cache the output of CGI scripts.

HTML forms provide a familiar and effective interface for eliciting user input, and you can use stylesheets to format them in a way that harmonizes with the rest of your page. (You can even animate the form elements if you want to, though this may not be a good idea.) However, there are other ways of invoking CGI scripts that may be more suitable for some sorts of application.

We mentioned in Chapter 14 that assigning to the `href` property of a window's `location` object has the effect of loading a new page. If we assign the URL of a CGI script, and append a query string, the new page will be constructed from the script's output in the usual way. The query string can be constructed by a script running on the client machine, on the basis of whatever logic is written into it, within the restrictions that must be applied to client-side scripts. Conventional form elements may not be required to obtain the necessary user input to construct the query; it may be sufficient to detect events on elements of the page, such as images that can be chosen with a mouse click. In a similar spirit, the additional scripting facilities introduced into Flash with the release of version 4.0 include the capability to send an HTTP `GET` or `POST` request, with a query string constructed by a script associated with elements of a Flash animation.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 542
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Beyond CGI

The CGI mechanism has two significant limitations. The first is that it can be slow, which makes it unsuitable for applications that may receive many hits. Various techniques have been and are being developed for improving performance while retaining the essence of the interface. A more fundamental limitation is that the output of a CGI script is returned to the client as an HTTP response. In many respects, this is actually a good thing. It means that any HTTP client can communicate with any CGI script, and that caching and security are handled automatically by HTTP. At the same time, it limits the way in which the client and script can interact to the possibilities supported by HTTP — a protocol explicitly designed for the efficient transport of hypermedia, which is not the only model of multimedia, as we have seen.

A major shortcoming of HTTP for performing complex transactions between clients and servers is the lack of any persistent state in the server. Even though HTTP 1.1 allows connections to be held open, the server still deals with each request in isolation. This extends to requests that invoke CGI scripts: each such request causes a new process to be started to run the script, and terminated once it has done its work.[16] In order to maintain continuity between a series of requests from the same user, various more or less underhand ruses have to be employed, such as embedding a connection key in a hidden form field in the script's output, using 'cookies' to remember state information on the client's machine (if the client accepts cookies), or using non-standard server features instead of CGI.

16
This is the main source of CGI scripts' inefficiency.

An alternative strategy is to bypass CGI and HTTP entirely, by writing a special-purpose server and client to implement a specific distributed application. This can be where Java comes into its own, not necessarily because of any virtue it may possess as a programming language, but because the libraries distributed with it provide extensive support for network programming that is (relatively) easy to use — opening a TCP connection and creating an input stream that enables you to read from it as if it were a file only takes a couple of lines of code, for example. UDP connections are also supported. Java also supports remote method invocation, meaning that a program can be truly distributed, with code running on a client calling methods on objects running on a server, in almost

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 543
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

the same way as it calls methods on local objects. This provides for seamless cooperation between the client and server.

In Chapter 14, we briefly described the range of facilities offered by Java's APIs for multimedia programming on the client side. It should be evident from that description and the preceding paragraphs that it is quite possible to construct distributed multimedia applications that combine media types — possibly including new or special-purpose ones that are not supported by mainstream HTTP clients — and protocols — again, possibly including ones that are not otherwise supported. The applet mechanism makes it possible to embed the functionality of such applications in the ubiquitous Web browser, and to allow users to download the client software when they need it. Alternatively, the reusable component technology known as Java Beans makes it possible to construct clients in a form that can be easily incorporated in other applications.

Where the computation in the server is mainly concerned with accessing databases, as it often is, the *Java Database Connectivity (JDBC)* API can be used. This provides platform-independent access to databases that use SQL as their query language. This is the vast majority of serious conventional relational databases. Although commercial database systems do not allow media data to be stored directly in databases, they can be used to store and retrieve data about such data. For example, an index to a collection of video clips, storing their title, length, the codec used to compress them, the date they were made, the director, and a summary of their contents could be stored as a set of relations. A URL at which it was stored could be used to represent each clip — being a string, a URL can be stored in a database where a QuickTime movie cannot. A client program could provide a user-friendly interface to such a database, and construct SQL queries from a user's input. A server could then execute these queries, via JDBC, and send the URLs of clips satisfying the user's criteria back to the client, which could then retrieve the clips and display them, using the facilities of the Java Media Framework or QuickTime for Java.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 544
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Further Information

[CHW99] is an expert summary of the protocols and related matters introduced in this chapter; [Kos98] is an accessible account of multicasting over the Internet. [MS99] includes much informed speculation about the future of the Internet, including the impact of multimedia. The full details of the major multimedia protocols can be found in their defining standards: [FGM+97] (HTTP), [SRL98] (RTSP) and [SCFJ96] (RTP).

Exercises

1. True or false: file transfer applications such as FTP can only run on top of a reliable protocol such as TCP?
2. Why is a sequence number on its own not sufficient to synchronize separate RTP streams?
3. Explain why TCP cannot be used as a multicast protocol.
4. Since the Internet is a network of networks, packets often travel over networks belonging to several different ISPs. Normally, ISPs operate 'peering agreements', whereby they provide their services free to other ISPs of roughly the same size, in return for reciprocal services. This way, ISPs don't have to keep track of where packets come from and go to, and charge each other for carrying them. Discuss how the commercial provision of multicasting may affect such arrangements.
5. The `HEAD` method of HTTP is much like `GET`, except that it causes the server only to return headers, and no body, in its response. What might this method be used for?
6. If RTSP is used to control a multicast stream, what would you expect to happen if a client sent a `PAUSE` request to the server? Discuss the implications of your answer.
7. Is it true that, as we claimed on page 517, 'There isn't much point in using URLs with fixed query strings in anchors'?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 545
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

8. RTSP allows the client to randomly access any frame in a video stream (provided it is not live) by specifying its SMPTE timecode. If you were designing an application that allowed users to control streamed video, what controls would you provide to give them this facility? Would your design task be any easier if additional information about the video stream could be transmitted separately? If so, what information would you use?

9. Video streams are often described as 'bursty': that is, the data rate is not constant, but exhibits bursts or intermittent peaks. Explain why this pattern occurs, and describe the problems it presents to networks.

10. The application outlined in the last paragraph of this chapter could largely be implemented using client-side scripting, CGI and HTTP. Describe how this could be done. What desirable features of such an application require you to step outside this technology?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 546
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Multimedia Practice

This chapter is different from the rest of the book, both in its form and in the nature of its content. It is entirely concerned with putting ideas into practice — with learning through doing — and is presented as a series of multimedia projects, preceded by some general advice and explanation. It is not intended that these projects should represent a general cross-section of real-world briefs — their purpose is to focus learning through solving a range of different problems. Each one has been designed to highlight certain areas of difficulty in practical multimedia, and to stimulate original and well thought out responses. Nevertheless, many of the briefs do draw their inspiration from existing or potentially real multimedia productions, or from actual problems we have been asked to solve in practice. There are no trick questions — the challenge lies only in planning or carrying out the projects themselves. In order to assist you in reaching appropriate and satisfying solutions, each project is discussed in some detail, and the main areas of difficulty are clearly identified.

We know that very different kinds of practical facilities will be available to different readers, and that the amount of time it will be possible for you to allocate to any project will also vary considerably. Some of the projects are small, some could be very ambitious indeed, and one or two could not actually be implemented with currently available technology. However, for each one, a variety of approaches is possible, and you should tailor your response to fit your personal situation. What you cannot execute in practice, you might still be able to plan fully on paper; where only some of the necessary media tools are available to you, you can try to improvise an alternative implementation of the brief with the facilities that you have got. However, some of the briefs have been limited with certain restrictions for a good reason, which will be

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 547
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

clearly indicated — to ignore the restrictions in these projects would be to miss the point. Even if you cannot do any practical work at all, much can be achieved simply by reading, thinking, and planning.

You may be working independently, or in a group — this too will determine how much you can achieve. Do not set out to do what is clearly impossible, but try to be realistic. To put ideas into practice successfully, you need to make an accurate assessment of what you, or your group, can actually produce. You will need to distinguish between the more ambitious and imaginative projects, and those briefs which are relatively simple and realizable. It is intended that some of the projects will function primarily to stimulate thought and problem-solving abilities, and these may be very difficult or impossible to actually carry through in practice. Others — the majority — could be realized without too much difficulty or the need for unusual equipment. If you are working in a group, there will be additional questions of project management and the division of labour to address, which will add a further challenge to the realization of each project. Groups need to agree upon an effective and productive organization of work, taking into account the personal strengths, weaknesses, skills and experience of group members. We do not offer guidance on specific management strategies, as each project is open to many possible approaches, both by individuals and by groups of varying sizes and abilities.

Inevitably, when it comes to actually creating a digital multimedia production, you are dealing with design: design of an interface; design of a coherent production; maybe even content design and creation. It is most important that you make an honest assessment of your own competence in this respect. Unless you have spent several years on a studio-based course in a design school, or have equivalent work-based experience, you need to recognize that you do not have the advantages and experience which others, properly trained in their field, are expected to have. Never be afraid to seek whatever help you can get from specialists — whether from fully-trained professionals or from people studying in appropriate areas. This is a fundamental part of good professional practice, and your work can only be the richer for adding other expertise to your own. However, if no such help is available, clear thinking, common sense, and a keen awareness of other people's work and your audience's needs or desires, can take you a long way. By studying both traditional and new media, and using the World Wide Web, you can gain unprecedented access to other people's work and ideas; if you analyse contemporary practice in individual media and multimedia, and examine design precedents carefully, you can build up a sound and informed basis for your own work. Much time and effort — not to mention disappointment — can be saved simply by learning from other people's successes and mistakes. It may sound like a piece of advice too obvious to be voiced, but it is so often overlooked, or scarcely acted upon, that it can hardly be repeated too often.

What is true for technical people who lack training in design, applies even more to multimedia designers lacking technical expertise. If you fall into this latter category,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 548
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

you too need to acknowledge your shortcomings, and to seek skilled help. But for you, if help is not readily available, a study of other people's work will not serve as a substitute. Too often we have seen multimedia presentations — especially Director productions — where inadequate technical knowledge has resulted in serious flaws in functionality, such as leaving users without a way of stopping a certain part of the presentation, or without any exit option at certain points. Where scripting or other programming skills, or highly technical knowledge, is necessary to make a presentation work, do not try to take a shortcut around this requirement. If it is impossible to obtain skilled assistance, you must either look for a solution to the problem which will fall comfortably within your own — or your group's — sphere of competence, or you must undertake to acquire the missing skills or knowledge at an adequate level.

Underlying principles and theory may be learnt from a book, such as this one — but to put this theory into good practice is another matter. Your multimedia work will be judged not by what you know, but by the practical results. Make yourself aware of the context in which you are working, make good use of specialist skills and your knowledge of precedents, and then *practise*. We all make many mistakes, and where practical work is involved these will quickly become apparent. Test out your designs and ideas at all stages — whether they are still in your head, planned on paper, or actually realized as multimedia projects. Avoid choosing your test audience from the people you believe will respond most favourably, but if you can, seek out impartial or unlikely test subjects — and then listen to what they say. You may ultimately dismiss some people's responses as being irrelevant to what you are doing, or you may decide that you are right and they are wrong, but make sure that you have both heard and understood their criticisms, and can give good reasons for dismissing those that you reject.

Any practical challenge offers scope for the imagination and for ingenuity. In some cases the result could be wildly fantastic, and in others a simple but elegant piece of classic design. The key to the successful realization of a project is never to pre-judge or copy, but to think through each task that has been set until you have reached a clear personal vision of your best solution. Then keep that vision in mind throughout all the stages of development, to use as a measure of your progress. Sometimes it will prove necessary to amend your original plan — so be it. The ability to adapt can be a key virtue in a designer — but do not lose sight of your goal, even though you may have to alter the route by which it is reached.

Rather than present some patent set of design rules or guidelines — a list of 'dos and don'ts' — we simply identify, within the descriptions of the projects, some of the more important issues you will need to address. So far as they concern layout, presentation and content, ideas of 'good' design are very much a matter of fashion and cultural sensitivity. It is true that a few classics of design may be widely admired and achieve a relatively long life, but on the whole, what looks wonderful in one decade (or even one

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 549
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

year) becomes out-moded, or positively unthinkable in the next,[1] and what may be the height of good taste and fashion for one group of people could be grossly offensive to another. Just be aware of what you are doing, and who you are doing it for. Even at a particular moment in time, and in a single place, many different standards of design will obtain among different social, economic and cultural groups. With multimedia productions often intended to reach a worldwide audience, the picture is extremely complex and unclear. It is highly probable that what seems good to you, or in your immediate social and cultural context, will most certainly seem bad to at least some part of a world audience. So, the wider the audience you wish to reach, the greater the challenge — but where you have a specific and restricted target audience, your task will be less difficult.

Interface design differs from other design elements within a project in so far as it is concerned with functionality and ergonomics. Some of the projects have a specific focus on interface design, and for these we have indicated some of the things you will need to think about — but remember that, even here, nothing is written in stone. It was not so very long ago that the now almost ubiquitous 'windows, icons and menus' interface, with its pointing and dragging, was unthought of. Before its invention — and for a considerable time afterwards — most computer users interacted with their machines in an utterly different way. Unless new designs depart from existing paradigms, there can be no significant progress. There is always room for good new ideas, and designers of interfaces — like designers of anything — are likely to continue to try to push forward and break new ground. As a general guide, it may be useful to bear in mind what Malcolm McCullough has expressed in the following way:[2]

> "Better interfaces relieve attention overload by getting computing out of the way ... changing it from active task into passive context. Digital production occurs within a highly visual and networked environment where our primary context is communication."

Usually, in the contemporary world of digital multimedia, our primary context is indeed communication, whether we are on the 'sending' or the 'receiving' end. It is no longer fashionable to say that "the medium is the message", but this is necessarily the case to some extent. If you wish to communicate *within* the medium as far as possible, you will need to minimize all aspects of that medium which are intrusive — which draw attention to the medium itself or to the business of interaction. For example, if you are displaying a list of product photos and descriptions for consumers to choose between on a Web page, it is unlikely that incorporating a large, jumping

[1] Fashion in clothing illustrates this most effectively — consider the history of flared trousers, or the bustle. But it applies equally to all aspects of design, from fonts to table-ware to billboards and TV advertisements.

[2] [McC96, p. 262]. This expresses very nicely a current state of thinking about computer use. Future developments will no doubt bring unpredictable changes to the way we feel about computing.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 550
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

and distorting animated title as a header to that list will best serve the intention of having potential customers focus on the products. Or, if it is not made clear in a piece of interactive multimedia what inputs from the user will give what results — or if it is difficult to achieve those inputs — your interface is itself presenting a challenge in interaction. This is fine if that is what you intended — for example in a game or certain types of educational product — but not otherwise.

Several recent books (for example, [LH99] and [EF99]) have proposed formal methodologies — derived more or less explicitly from current practice in software engineering — for multimedia development. If you are familiar with software engineering practice, you may find it helpful to try and apply the same principles to your multimedia practice, but you should bear in mind that there are fundamental differences — centring on content — between the two disciplines. Multimedia is a new area, even by the standards of computing technology, and we still lack a sufficient understanding of its nature to be able to quantify the processes of production and reduce them to a set of rules and procedures.

In thinking about and carrying out the projects described in the remainder of this chapter, you should feel free to ignore any of the advice we have just given. This is a chance to explore multimedia production in a safe context, free from the pressures of deadlines, clients, and the test of public opinion. Hopefully, the experience you gain will help you avoid some of the worst pitfalls when you are exposed to those pressures on real projects.

Note that the projects are not presented in any meaningful order. This is partly because what presents a big challenge to some readers will be an easier task for others, so ordering according to levels of difficulty is not really appropriate. (The one exception to this is the final project, which represents the culmination of the whole book.) And partly because while several projects may deal with similar media areas — they may share a focus on sound, for example — they will also be concerned with quite different issues in each case. So you may approach the following tasks in either a linear or non-linear way, according to your own preference. It really doesn't matter which.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 551
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

The Projects

Choose a personal hero or heroine — living or dead, famous or unrecognized, real or imaginary, human or non-human — and design a *single* Web page (i.e. one without links), using whatever media and software are available to you, to present your chosen figure in the best light to a worldwide audience. Make the design of your Web page 'fit' your hero or heroine in every possible respect, and keep the total size of the content below 250 kbytes. (If you want to make this a bit harder, keep the content down to 100 kbytes.)

This project is about effective communication in a very small space, and about sensitivity in design. You will need to be highly selective about the material you choose, process it sufficiently to reduce its size to meet the target, and design and organize your page in an appropriate way. In asking you to make the design 'fit' the figure you have chosen, we mean that you should choose and integrate all the elements of the page in a way that is sensitive to your subject. For example, if you choose a great political or religious leader, you might wish to encourage viewers in a feeling of respect and admiration for their achievements or philosphy — an irreverent comic strip or a quirky font would be out of place. If you were to choose a film star, on the other hand, a touch of glamour and a showbiz aura for the whole page might not go amiss ...unless, perhaps, you wanted to convey their personal strength in some aspect of life unconnected with the film industry. If you were to choose your pet dog, or Bugs Bunny, a completely different solution would be appropriate. So think hard about exactly what it is that you are trying to communicate, and then make sure that every aspect of your design works towards that end. Be aware of the difficulties in addressing a world audience effectively, and think about ways of getting your message across to as many different kinds of people as possible. Do not be discouraged if you have only limited technical resources — results can often be more effective when the means are simple. If you do have substantial resources, excercise discretion in their use, concentrating primarily on effective communication within the limits set. Don't forget that less is quite often more.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 552
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Traditionally, museums and galleries prevent visitors from approaching valuable objects or works of art too closely. Many items are protected by glass cases or barriers, and very delicate objects may be displayed under low lighting conditions. It is almost never possible to see the objects displayed in any context or arrangement other than the one ordained by the exhibition organizers.
Design a simple Web site or a CD-ROM, which exhibits a number of artefacts of your choice, and allows the user to view and examine the items as they might wish.

This brief may seem straightforward — there is no particular difficulty in setting up a virtual exhibition. The challenge lies in going beyond the confines of the exhibition paradigm. Imagine what visitors might like to do with paintings, if they were allowed to: take them off the wall and look at the back, examine them very closely, or from a distance, perhaps judge the effect of looking at them upside down. What about an exhibition of early Renaissance paintings, or aboriginal cave paintings, which are known to cover earlier paintings lying beneath? And different artefacts raise different possibilities. What might you want to do in an exhibition of ancient Egyption artefacts, or a display of stained glass window panels? It is your task to enable the virtual museum visitor to control the way in which they view and interact with the virtual objects to the greatest extent possible. Note that the emphasis here is strictly on the visual — the intention is not so much to inform as to provide a wide, user-defined range of visual interaction with the objects on display.

You will need to design an interface which allows satisfying displays of the objects while providing a simple and effective way of interacting with those displays to change them in a number of different ways. As direct interaction with the artefacts on display is the primary focus of this project, the business of having to interact with the computer as an intermediary will be intrusive, and your interface should minimize this disturbance as far as possible.

More ambitious solutions to this project will require considerable expertise and a wide range of software. So, be aware of what it is not possible for you to achieve — and therefore of areas which you should steer clear of.

Clearly, if you have the necessary software and the skill to use it, one fruitful approach to this project could be based on 3-D representations and VR.

Devise a multimedia application to assist in the identification of birds in a specific geographical area. This should use still images, sound and text — and video if possible (but if not, do without it). It should include an interface that allows the user to obtain, for example, a view of typical tail twitching behaviour, or a sample of warbling song, as well as stock responses to form-filling.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 553
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

The aim of this project is to make the end product serve a precise requirement in the best possible way. The requirement is for an aid to the *identification* of birds — not a catalogue or other attractive presentation. Birds are usually identified by information such as size, colouring and markings, voice, habitat, behaviour, and so on. You need to decide (or research) what features will aid identification, and then assist the user both in recognizing these features and matching them to the birds of your chosen area. Try to think through the ideal solution to this brief from the ground up — you may learn by looking at similar multimedia applications which already exist, but avoid copying them.

An artist wishes to convert a many-layered screen print into a multimedia gallery presentation. The original printed image consists of 24 layers, which will be scanned in separately. Each layer contains one coloured abstract shape or mark; some are opaque and some are semi-transparent. In the presentation — on a computer screen installed in an art gallery — these shapes and marks are to move around in an apparently random manner, independently of one another, giving an almost infinite possible number of permutations of the image. The ultimate purpose of this display is to enable the viewer to halt the movement at any time, and request a printed version of the particular permutation of the image which is displayed at that moment.

This is another brief which appears deceptively simple: several of the technologies described in earlier chapters, including Flash, Director, and JavaScript, will enable you to animate layers. However, you must ensure that the technology you choose does justice to the material. Does it handle transparency correctly, for example? There is also an interesting problem in deciding what constitutes random motion of the layers. Presumably, an unpredictable swirling effect is required. How will you achieve this? And if the user's reactions are a little too slow, such that they stop the movement beyond the state which they wished to freeze, how will you ensure that they can retrieve the image they actually wanted?

The real difficulty of the project is concealed in the innocuous final requirement that the user be able to request a printed version of the current display. This is a fine art project, so screen resolution prints will not be acceptable. Accurate colour is important, both on-screen and when printed. How will you arrange that a high resolution image can be printed that is identical to what is displayed on the screen at the instant the print is requested?

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 554
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

An old harbour town has decided to mount an exciting multimedia installation as a tourist attraction — to be sited in a boat moored by the waterfront. Inside the boat, visitors are to be given access to a vast amount of information and material in all media on the subject of 'harbours through the ages'. Devise an effective and realistic implementation of this installation, with an interface suitable for constant use by the general public, and some measure of control which will prevent the visitor from being overwhelmed by excess material or getting disoriented.

The first problem here is obtaining the material. A possible approach is to hook up the boat to the Internet, but a conventional Web browser is too general a tool. You might write a dedicated Web client that automatically queried the major search engines at intervals and then provided a set of links for users to follow (but how would you prevent them getting lost in hyperspace, or stepping outside the material on harbours?). Alternatively, you could download the material, and convert it to some standalone form of multimedia production. How would you store, organize and retrieve the material — and make sure it stays up to date — in this case? As a third alternative, consider how you would obtain suitable material from conventional sources, and digitize it. Make some estimates of the costs of this approach, and consider the problems of copyright and clearances.

The second challenge lies in designing a suitable interface for use by people of all ages and abilities. This is a classic design problem, without an emphasis on special needs. For this particular project it would be a bonus if the interface could relate to the subject of the installation (harbours) in some way.

Construct a Web site for use by blind and severely visually handicapped people who are computer literate. The site is to provide a guide, tailored to the needs and interests of the visually handicapped, to a city of your choice.

In one sense, this is easy. The W^3C accessibility guidelines, which ought to be compulsory reading for all Web designers, describe how to use the features of HTML 4.0 to design pages that can be appreciated by blind and visually handicapped people. So this is partly an exercise in applying the guidelines — but technical fixes are not the whole solution. You need to think this through for yourself — to choose appropriate material, and organize it in a way that does not depend upon the visual character of a Web page. If possible, consult somebody with visual difficulties before you start, and then have them evaluate the site you produce to see how well you have appreciated their problems and needs.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 555
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Create a simple Web site which contains basic but perhaps rather dull information (or use an existing one), but add to it all that is necessary for the user to build and experience their own sound environment (within reasonable limits) to enhance their experience of using the site. This sound environment should not relate to the other content of the Web site in any way — its function is solely to make the experience of using the site more pleasant.

We observed in Chapter 12 that sound can be particularly irritating, and that it is a bad idea to add music or other sound to a Web site unless a user can turn it off. In this project you are invited to go further, and allow the user to select background sounds that they like to accompany their browsing of this rather dull site. There are three sub-problems: providing the sound, constructing an interface, and playing the sound. For the first step, don't feel restricted to music; natural sounds, such as running water or rustling leaves, might have more appeal. This project offers you the opportunity to experiment with making sound recordings and digitizing them. The interface could simply allow the user to preview sounds, and select one to play in a loop or for a certain time, or it could allow for sounds to be mixed or altered to create a 'soundscape'. The latter will require programming, but if you cannot program, just designing the interface offers a challenge in itself. You will need to consider how the sound is to be delivered from the server to the client. Solving these problems should determine how you can implement the actual realization of the sound environment.

Create a CD-ROM and/or a Web site guide to an art college's annual degree show. This needs to display a wide range of work in such disciplines as glass, ceramics, fashion design, theatre design, graphics, illustration, photography, video, animation, painting and drawing, sculpture, etc. It should convey the scale of work, and the spaces in which it is displayed in the real show, where that is appropriate, and it should allow for each student to include a written statement and short personal profile to accompany their degree display. Your production should provide adequate navigational controls for the user to find their way around both the virtual and the real degree show.

This might almost be called a classical problem in multimedia, so commonly is it required. Remember that, in this case, the multimedia production is subsidiary to the real show. Although it should be attactive and useful in its own right, its primary function is to advertise the degree show and serve as a guide to it. Thus you should, in some way or other, represent the physical space of the show, whether as a map or using QTVR or some other 3-D representation. You must also respect the students' work. Where it is represented, you should do it justice, and not subsume it to some grand design scheme of your own. Different types of work — painting, sculpture,

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 556
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

video, and so on — need different treatment to convey their nature. Do not fall into the trap of trying to produce a solution that will work for any show. Fit your solution to specific work.

The best results will be produced if you are able to work with an actual art college or faculty on real students' work.

A chain of real estate agents wishes to provide computers in each of their branch offices which will access a central database holding pictures, video and VR material relating to the properties for sale. Devise such a system — using the Internet and not a local network — which allows for this centrally held material to be pulled down for display to walk-in clients.

This brief is representative of a class of multimedia applications that are being deployed in the world, although at present, intranets are usually used and not the Internet. It is a large-scale, technical project, and should only really be attempted by a team that includes a high proportion of technically skilled people. Its importance lies in the need for design and computation both on the server and client side. On the server, it will be necessary to design and implement a database that can hold (or at least point to) multimedia data. On the client side, an interface that can be used by office staff to display property details in a suitably impressive manner to clients is required. The client and server must communicate. HTTP and CGI may be adequate with a conventional Web browser interface, but you may be able to design a better (at least a more efficient) solution.

Produce an interesting and enticing presentation of life on your campus which uses just recorded and remixed sound and still images (no video), to play as a looped show on a simple fixed installation without interactivity, in a foyer or waiting room where potential students are waiting to be interviewed.

This is an exercise in production and communication skills using relatively simple means. The result will essentially be a looped slide show. To compensate for this simplicity and to avoid possible boredom on the part of the viewers, make your production as imaginative and interesting as possible. Avoid the obvious: consider using devices such as humorous or ironic juxtapositions of pictures and/or sound, and consider how to achieve emphatic effects. Think about the pace and flow of the production — try to create variety and dynamic relationships between elements of your material. Make the most of the media you are working with — still images could come from many different types of sources and may be treated or combined digitally. Sound alone can be extremely powerful, so make the most of its potential.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 557
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

A maker of short animated films wishes to give the audience a new level of interaction with her work. A ten-minute film has been made frame by frame with traditional animation techniques, and captured to computer disk, complete with sound track. The film-maker would like the audience to understand more about how the film was made, and to be able to view individual frames or selected sequences at will. More challengingly, she would like any viewer to be able to re-edit the film, and play back their own edited version. This version would persist until either the computer was instructed to return to playing the original version, or another re-edit took place. Devise an implementation of this brief which would be suitable for installation in a public space, with the display of the film versions maintained on a TV monitor or video projector separate from the computer interface.

It's hard to see how this can be implemented without some custom programming. You might consider using one of the consumer-oriented desktop video editors, such as Apple's iMovie, but before doing so, you should think very carefully about whether an untrained audience could make use of it, without getting into difficulties. If you prefer a custom solution, you will need to devise a minimal interface that lets the user perform enough re-editing to obtain some feeling of control, but is simple enough to be mastered without instruction. (Non-programmers can usefully try to design such an interface without implementing it.)

The technically minded should also consider the problem of how to maintain the display of one version of the movie while somebody is creating a new edit.

Create a very simple, low-budget presentation of a set of instructions of your choice — for example, the best way to brush your teeth, or how to give basic first-aid. Assume that this presentation may have to be installed on a very old computer. The instructions are to be text-based, with links via clear and appropriate icons to a pure graphic exposition of the same instructions, for those who, for whatever reason, cannot read the text.

This apparently simple project is not easy to do well. The challenge is to communicate absolutely clearly, in either text *or* pictures, but not in both at the same time. The design of the presentation should also promote clarity, both in the general layout and ordering and in the design and placing of the icons for the links. Think about the presentation of material one screen at a time. If you do not have graphic design or illustration skills, you could usefully collaborate with someone who does for this project, as the purely graphic exposition of instructional material presents a substantial challenge in itself. However, if this isn't possible, you could use photographs or existing material (providing you observe copyright restrictions).

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 558
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Remember that the brief specifies that this production must be capable of running on a very old and basic system — don't deny information to people who do not have access to the latest and most powerful technology.

Imagine a portable tourist's companion of the future. When turned on, this hand-held digital device will: tell you where you are in the world (or galaxy); ask you what you would like to know; show you appropriate multimedia material; give you directions on how to get where you want to be from where you are now; make reservations for travel, accommodation, etc. on your behalf; and connect to Web sites for further information where appropriate. It will update itself to remove redundancy, and hold a memory of your personal use for user-sensitive assistance. Specify a design for such a system, and if you feel you have the skills and necessary facilities, implement such parts of it as you can.

Which may be none of it... This is an ambitious project that goes beyond the capabilities of today's technology (doesn't it?). This should not prevent you from thinking about what is required, from hardware, software and telecommunications. A possible starting point would be to look at how nearly you can approximate this device with the facilities currently available: hand-held PDAs, mobile phones, and the Internet. It may not be too big a technological step from there to the full system. However, you should also think about what is required beyond the technology: Who is going to provide the information? How is it going to be paid for?

You may find it possible to implement some of the sub-tasks; if so, try to combine as many as possible into a single system that approximates the intended implementation. Make an honest assessment of how nearly your prototype approaches a genuine realization of the brief. (Try to see this from an ordinary consumer's point of view.)

Thinking about how hypertext, displays, frames, etc. work, design a layout for an on-screen presentation of a substantial and heavily annotated manuscript such as the Bible, the Koran, the Buddhist Sutras, the complete works of Shakespeare, or whatever.

Footnotes and endnotes, as we know them, are a device entirely dictated by the physical form of books and the process of printing. Digital media frees us from those forms, but what are we to do instead? Is putting an HTML anchor node wherever you would place a footnote marker any more convenient than putting numbered notes at the end of the text? The aim of this brief is to utilize the dynamic nature of screen layout to devise a mechanism for associating annotations with a text — a mechanism that is convenient, placing the annotation where it is needed, but unobtrusive. It

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 559
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

is a good discipline to try and work within the limits of Web technology (you will probably need to write some scripts), and this may allow you to produce a prototype. You could also think about ways of automatically generating texts in your layout from XML markup.

Create an instructional road safety installation specifically for teaching blind or severely visually impaired children how to interpret sounds at the roadside. This should incorporate some very simple interaction with an appropriate interface for these users — to allow, for example, for repetition of elements, starting and stopping the instructions, and so on.

This is not simply a re-run of the earlier project for blind people. Here, sound is a vital part of the system, but must be interpreted. What is required is not a guide, or a collection of information, but a complete learning environment. This offers an opportunity for you to research and think about educational uses of multimedia, as well as designing both for children, and for special needs. The interface does not just duplicate the familiar functions of a Web browser, but has to provide special controls suited to the teaching purpose of the installation. Flash [3] may be a suitable tool for implementing at least a prototype.

Import a short (say five minutes or so) but already edited piece of video or an animation into a video editing application. Study it carefully and choose 20 or 30 single frames which you feel will best convey the narrative or import of the whole clip on their own. Produce these as (a) a storyboard, and/or (b) a time-based animatic.[4]

This is an exercise in reverse storyboarding. Its purpose is to make you think hard about how time-based visual media are constructed and develop, and to enable you to identify and subsequently plan structures and narrative development for such work of your own. The challenge lies in identifying a few key frames out of some 9000 candidates. You should test out your result on someone who has not seen the original piece you started from — your success will be measured by how accurately they are able to understand the story or development of the piece from your storyboard or animatic.

[3] You will need Flash 4 for the extended scripting features.

[4] An animatic is a sort of sketch of a movie composed entirely of still images, each held for the duration of the scene envisaged for the final, full-motion, movie. Each still image should capture the essence of a scene. The animatic will show the rhythm of the edits that are proposed (or, in this case, have been made) and the flow of the whole movie.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 560
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Choose a popular computer game, and carry out a reverse storyboarding process similar to the one described above — this time for a non-linear structure — to convey the essence and possible developments of the game to someone who doesn't know it. This project may be carried out either on paper — in which case devise a suitable non-linear structure, or in a digital form. The number of elements in your non-linear storyboard will be determined by the game you choose, but you should keep them to the minimum possible.

Carrying out the brief should not, in itself, present too many problems. To learn from this project you should ask yourself how well the storyboard manages to convey the non-linear structure, compared with the way it worked for the linear structure in the previous project. Can you devise an alternative representation that performs the same function but is better suited to non-linearity? Given the non-linear nature of much multimedia delivery, the ability to visualize, plan and represent non-linear structures is of fundamental importance. Is it appropriate to develop a personal way of representing such structures? Do you think the problems and constraints will be different or broadly similar for each non-linear project you might undertake? These issues are fundamental to the production of non-linear multimedia productions at the present time.

Why would it be pointless to try to construct an animatic for such a structure? Can you think of any way of producing a time-based representation of a non-linear production that takes account of user-driven events?

In Chapter 1, we invited you to choose a Web site and identify at least five ways in which you thought it could be improved. Return to your chosen site and redesign it, using its existing content, so that it is as good as you can make it.

This ought to be a simple project by now. You should be able to determine which of your original suggestions for improvement are implementable, and implement them. It may be that, having now learned more about multimedia technology, you must accept that some of the site's features which you found inadequate are unavoidable consequences of Web technology. For these, consider ways of changing the site to reduce their impact. In order to make your changes, you should either rewrite the HTML by hand, or import the site into an authoring application and change it there.

The original exercise in Chapter 1 also asked you to criticize a CD-ROM. You may like to consider, for this case too, whether your original suggestions for improvement can be implemented, and how.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 561
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Design an interface — and make a mock-up if appropriate tools are available to you — for an international tele-conference for students to exchange thoughts on their experiences of multimedia courses.

Think about what you need to control; what you need to indicate and the best means of doing this; what analogies you should be thinking in terms of (is this to be like using the phone, watching TV, being in a conference centre, or what?); and what assumptions about the underlying technologies you are or should be making.

There are several tele-conferencing applications in use, and you should look at their interfaces as a starting point. None of those that we have seen give the impression of having been designed — each participant is put in his or her own window, equipped with a few controls. Try to think of an interface that fits the application, not the desktop context in which it will be used. One possibility is to use a three-dimensional representation of the conference.

Create for yourself a show-piece Web site which you could use to exhibit your design skills to a potential employer or client. This should be a no-budget production (except for using such equipment as is available to you) — the content should be entirely comprised of royalty free material trawled from the Web.

The purpose of this project is to help you to understand and maximize your own particular Web design skills, without being concerned with the production of content. However, you should ensure that you choose and organize your found content in such a way as to create a successful and coherent Web site. Make use of the full range of Web technologies if you wish, but ensure that your page degrades gracefully on older browsers, or when viewed over a slow link. And be careful to test it on as many different platforms as you can — you won't impress many clients if the first thing they see is a scripting error message.

Using the material from your college's Web site (or a part of it), make a Director movie that conveys the same messages and information, but more effectively and in a way that is better suited for distribution on CD-ROM. Add appropriate supplementary material if you can, to take advantage of the change of delivery medium.

The rise of the Internet as a delivery medium means that more and more content is being 'repurposed' for the World Wide Web. In this project, we are asking you to go in the opposite direction, in order to gain an insight into the different requirements

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 562
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

and opportunities of online and offline delivery. You could take the exercise further. For example, you could produce a print version, using text and images only. At each stage, see how the form must change to fit in with the medium. Alternatively, try to avoid changing anything, and see whether the form and content used in one medium fit comfortably in another.

This project should be done in groups. Each of you should obtain or make one or more pieces of text, audio, video, animation, and one or more still images — you each need a total number of media elements sufficient to send a different one to each of the other members of the group. (So if your group has ten members, for example, you each need to collect and send nine media elements.) Each group member will end up with a collection of elements, with no particular logic to their combination. The task is to create a multimedia production out of those elements, first by working individually, and subsequently by combining your individual productions into a larger whole.

It is important for group members not to collaborate beforehand; the idea is that each of you is presented with a collection of elements over which you have no prior control. It may be that one member ends up with nothing but pieces of text, or a large collection of sound files, or things may work out so that everybody ends up with a balanced number of elements of each type. The point is to make the best of whatever turns up. This means looking for ways of using the particular mix of media that you have been sent, looking for connections or contrasts between media originating from different people, and so on.

This project can be carried out on a small scale — within your particular class, for example — or on a much larger scale, by collaborating with other groups around the world. You can choose whether or not to focus it around a specific subject, or to leave it wide open (which is more difficult to do well).

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 563
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Malcolm McCullough suggests "we might say that postmodern consumers are ceasing to spin their own yarns, figuratively, every bit as much as the artisans of industrializing Britain stopped spinning their own yarns, literally."[5]
Thinking again of the dark and stormy night in the Introduction to this book, design an exciting and enabling CD-ROM or Web site which will allow postmodern consumers to spin their own yarns once again, with the only constraint being that the yarn commences with the expression (not necessarily in words) "It was a dark and stormy night ...".

This is one to dream about and plan; it is not expected that you will attempt it in practice — unless you really feel that you are in a position to do so. This is not about providing a ready-made story with alternative branches — or anything like that. It is about trying to invent a mew multimedia form for the future. What would be required from a multimedia production that enabled each user to spin a yarn, using all possible combinations of media, which was completely of their own making but from a single starting point? Give free scope to your imagination, but temper it with some hard thinking about what is required for this to become possible. Somebody has to lead the way ahead ...

[5] [McC96, p. 73].

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 564
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Afterword

The Creative Challenge

Brent MacGregor
Head of School of Visual Communication, Edinburgh College of Art

Multimedia has arrived. Developments in hardware and software now make it possible for things only dreamt of a few short years ago to be created easily. Domestic machines can play complex multimedia product and easy to use software tools allow the creation of imaginative work on the domestic desktop. Once created, such products can be recorded cheaply onto silver discs or instantly published on the Internet. The technology is now enabling, not inhibiting; it has delivered its promise. What is needed now is for a creative infrastructure to develop alongside the software and hardware. New creative dreams will be dreamt and realised as multimedia matures and develops in the hands of a wide range of imaginative people. This revolution is only beginning.

User base

I remember an old joke from the early days of multimedia. A developer arrives at the pearly gates having received the final error message. 'Who are you and what have you done in your life?' asks a holy personage. 'I created multimedia' comes the proud reply. St. Peter answers 'That's nothing, God created the world in six days.' 'But He didn't have to worry about an installed user base,' comes the frustrated response.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 565
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

I remember a time, not so long ago, when the phrase 'multimedia computer' was used to describe a PC with a double speed CD-ROM and a sound card. Multimedia developers worked in niche markets where the required hardware for consumption could be afforded. Today every sub-$1000 PC is multimedia capable with a fast CD-ROM if not DVD player, sound and video cards. Complex multimedia products can be used on ordinary mass market computers. The once desired user base has arrived. What is needed is the creative infrastructure to complement this 'kit'.

Just when every PC is shipped with a CD-ROM and several free multimedia products; when the physical manufacture of silver discs is cheap, the world has changed and DVD has arrived with its increased and much desired capacity. Will there will be another time lag as the installed user base arrives sufficient to justify huge investment in newly developed product? Fortunately, as with VHS, the predominant software for the DVD revolution already exists in the form of films originally produced for the cinema. This new technology offers high quality domestic reproduction of movies which may assist take up rates, but will the DVD player sit by the television in the living room or be part of the computer on the desktop or both? Will DVD offer unlimited creative possibilities, or will it just become another way of delivering Hollywood product?

Creative possibilities

These are just quibbles. It really is time to start getting excited about what is now or very soon possible. The printed book dates from the fifteenth century. Photography was developed in the 1830s, while film was invented in 1896. Radio programmes developed during the 1920s and by the mid-1930s television programmes were routinely possible. All these forms have been available to domestic consumers using easily affordable equipment for some time. In short, they are everyday commonplaces. The creative content growing from the technologies in question is mature but none have had their full creative potential realised let alone exhausted. All have been refined and developed as creative forms since the technology made them possible. Will multimedia be the same and develop over years, decades and even centuries or will it go the way of the telegram, telex or the newsreel and become a technologically enabled form quickly eclipsed by subsequent developments? Will the CD-ROM and its silver successors become the 'new papyrus'[1] or just a footnote in the history of creative forms? Will the development of multimedia forms and the growth of broadband network distribution be a creative revolution, a Gutenburg shift or just more noise? The answer to this question lies in the hands of creative people in design studios, in garages, in University computer rooms, in front of screens all over the world.

[1] *Multimedia: The Complete Guide*, London: Dorling Kindersley, 1998, p.8.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 566
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

A common sense description of multimedia might be the ability to combine the creative possibilities of radio and television programmes, newspapers, books, magazines, comic books, animated films and music CDs into one set of computer files accessed by the same piece of software to provide an integrated seamless experience where user input to some extent determines the manner in which the material is accessed. It is therefore interactive. The computer's ability to have rapid random access to the files which constitute this material makes the linear model of the radio or television programme seems old fashioned and limited. Interactivity (over and above the level of interactivity involved in the very act of reading or viewing a fixed, linear text) wherein the user to some extent determines the text or, more accurately, the order in which the text unfolds, offers great creative potential. It also offers a great creative challenge with the notion of an interactive movie very likely to give the traditional scriptwriter something approaching a migraine headache to the power of 5.

Multimedia already means different things to different types of creative people. Film and television programme makers quickly saw the possibility for interactive narratives with the arrival of the laser disc in the 1980s. Yet interestingly there are no great examples of this form to point to as creative landmarks. Graphic designers who began to work with computers to create two dimensional printed work, saw the ability to animate pages, to add sound and video as revolutionary extension of what they do and they found in the World Wide Web a distribution medium undreamt of. Fine artists have enthusiastically embraced the technology (Laurie Anderson, *Puppet Motel*, 1995[2] and Zoe Beloff, *Beyond*, 1996[3]) and will continue to act as the leading edge of development in a way we may not understand the significance of for many years. Animators often respond to the vision of multimedia by saying 'We've been doing this for years'. The only difference arising with the new technology is random access/interactivity. In fact, there is a case to be made that animation is the most helpful paradigm to apply as we seek to understand and create multimedia from the technical possibilities offered. It is the closest analogue analogue.

Interactivity

Traditional animation and film and television has a linear text unfolding in only one way, one narrative order, that determined by the creator. With the arrival of multimedia new possibilities have come to the fore: first hypertext and now hypermedia. This important notion presents the greatest creative challenge since the arrival not of printing but of writing. The move from linear analogue forms to digital non-linear creative structures will be a gradual one driven by organic imaginative growth over relatively long periods of time. New multimedia forms will grow, as all

[2] http://voyager.learntech.com/cdrom/catalogpage.cgi?puppet
[3] http://www.users.interport.net/~zoe/

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 567
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

new media do, from their immediate predecessors. Television drama grew from its radio, cinema and theatre origins over a period of decades but is now a creative form itself.

Old wine in new bottles?

Digital tools are now routinely used to create traditional linear product. The ubiquitous word processor gives the writer increased non-linear manipulative possibilities in construction but the product that emerges from this process is an orthodox text consumed, as the author intended, in a defined linear way. A similar process occurs in film, video and animation production where digital 'nonlinear' editors are now used to create traditional linear narratives. High quality contemporary audio visual product is shot on film for its still superior image quality and contrast ratio, but this film is digitised and edited 'non-linear' because of the creative flexibility and speed afforded while exhibition is still on film because of the quality and reliability. It is a case of the best horse for the course. Sitting in front of a computer screen is clearly not the best way to read *War and Peace*, which is doubtless best enjoyed as a printed book but teaching children to read using a speaking multimedia book running on a computer which records their progress adds value and creates a new order of experience.

The encyclopaedia, whether printed book(s) or silver disc, has always been interactive. Its contents are accessed as the user wishes, in a different order for each user with certain devices to help access. The content of the traditional printed reference work consisted of text and images. The digital multimedia version can add sound, it can animate the illustrations and add full motion video. Its search tools can be incredibly sophisticated and its content can be arranged and presented in a variety of ways. Crucially however, it still needs a huge team to produce the content. This content is fundamental and its creation requires a vast repertoire of traditional production skills ranging from making phone calls through organizing and managing teams to new skills such as cyberspace architecture. In this regard multimedia production is most like movie making, a team endeavour where technical specialists from best boy through gaffer and re-recordist are orchestrated by production managers and producers to realise the creative vision of scriptwriters and directors. In multimedia these roles are still evolving.

If a creative infrastructure is to develop to unleash the potential of multimedia, one must develop the equivalent skills of the producers, directors and the script writers who toil in the television and film industry. It is worth making the point here that television as it exists is already mixed media. The electronically transmitted moving pictures which define television are routinely mixed with voice and music, stills, text and graphics. In fact the only thing really missing for making the mixed media of present day television into multimedia is the element of interactivity. The viewer sits before the screen and television flows past in all its ephemeral glory. Even if you

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 568
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

record it, you can only spin backwards and forwards very ponderously without being able to access additional audio-visual material or shift from motion video to text. Television then already has all the elements of multimedia save the ability to view the material in any order the viewer chooses. True interactivity is still missing.

The role of the graphic design profession in multimedia development should not be underestimated. The typographical knowledge and skills, the sense of page layout applied to screen design and the ability to communicate concisely in visual language are all skills which are essential to the creation of any multimedia product. Multimedia screens or web pages all need this creative skills if they are to look anything but amateur.

Taxonomy of multimedia

The technical problems have been solved for the most part, creative challenge remains great. What is likely to be created can be seen by a brief taxonomy of existing multimedia products. From these cave paintings of the multimedia age we may be able to glimpse the mature forms that will emerge.

As mentioned above, the reference book, be it the dictionary or encyclopaedia, has been an obvious type of product to produce in multimedia format. In these early days, the interactivity offered by multimedia seems to work best if delivered using the book model which has always been both linear and interactive. The multimedia encyclopaedia is already with us. In the reference book you have an ideal multimedia product which allows for interactivity, indeed demands it. However, a multimedia atlas will never replace the printed version if the maps are not very good and the production of high quality maps is not a task to be undertaken lightly. A searchable illustrated multimedia pronouncing dictionary adds value but does it add enough to lure the consumer to part with cash? Virtual museums and galleries can give us access to images and objects physically distant and multimedia technology can be used in the same physical museum space to enhance the actual physical exhibition.

Education is an area where the potential for development is enormous especially as the young are the most likely adopters of new digitally enable products. 'Living books', children's stories which read to their user and help them to learn to read in the process are already among us. Mathematics and drawing packages already fill Christmas stockings put there by dutiful parents acting out of the same noble motives that used to keep door to door encyclopaedia salesmen in business. Languages can be taught with multimedia methods and huge, entertaining databases can make all the sciences accessible and exciting.

Computer games be they action adventure, puzzle or simulation already generate more income than Hollywood films and have the added benefit of creating a

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 569
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

generation of enthusiastic users unafraid of technology. Some of these end users will go on to become sophisticated creators of multimedia product.

However, in terms of adult product for the mature, non-games playing male and female audience there is still a golden opportunity to produce a mass market product. Interactive narrative is often mentioned in this context but this promise remains unfulfilled since it was first made possible with the laser disc in the 1980s. Simply stated there has been no 'killer application' in multimedia. One can talk of software products or games which define an era but there are no multimedia equivalents, no paradigm creating breakthroughs. The form is more difficult both to create and consume, it is demanding but it equally has more potential. The challenge remains; the opportunity is still there.

Conclusion

We live in interesting times, times when the young and the creative are taking the new digital technologies and appropriating them to their own uses; uses that could never have been imagined by those who developed the technologies. To work today in the creative uses of digital media is akin to what it must have been like to work in the film business with Eisenstein or Griffith as they defined the working practices and the very grammar of the film form, or what is was like to have been working in television during the fifties and early 1960s when that form defined itself creatively as a distinct medium different from both radio and the cinema. The situation is all the more complicated today as technologies very often become obsolete before they can be fully understood and used. Whatever the technology and however great the pace of change and development, however much the unexpected happens, analogue creativity must inform digital technologies. We who use any communications media whether new or old, digital or analogue: we still must have something interesting and worthwhile to say. A mastery of the medium and its techniques is crucial but so is vision and creative daring, imagination and flair and the ability to work hard to make visions real.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 570
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Annotated Bibliography

The references that follow have been selected as suitable sources of additional information, or to provide useful background. Since this is an introductory text, we have concentrated on books and tutorial articles, and have made no attempt to provide a guide to the research literature.

Because of the changing and ephemeral nature of the World Wide Web, we have only included a few Web pages among our references. For the most part, these contain standards documents, on the assumption that their location is likely to remain fixed for some time. The Web site that accompanies this book, at `www.wiley.com/digital_multimedia`, includes an extensive collection of links to relevant Web sites.

We have adapted the convention used for asides in the body of the text to indicate the nature of some of the references. Where the annotation begins with ⇨, the book in question is only tangentially relevant; where it begins with ⊃, its content is of a technical nature, and will require some programming or mathematical expertise.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 571
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[Ado90a] Adobe Systems Incorporated. *Adobe Type 1 Font Format.* Addison-Wesley, 1990.

This specification provides the information on how PostScript fonts are stored, and what hints can be provided by the font designer.

[Ado90b] Adobe Systems Incorporated. *PostScript Language Reference Manual.* Addison-Wesley, 2nd edition, 1990.

Most of the time, most people don't want to know what is going on inside a PostScript file, but for the times when you do, the answers are all here. While this book is the specification, there are also more accessible introductions to using PostScript if you ever need to.

[Ado93] Adobe Systems Incorporated. *Portable Document Format Reference Manual.* Addison-Wesley, 1993.

In case you ever wondered what PDF is actually like.

[Ado97] Designing multiple master typefaces. Adobe Technical Note 5091, Adobe Systems Incorporated, 1997.

This note contains an overview of the multiple master technology and guidelines for designing font families based on it.

[App93a] Apple Computer Inc. *Demystifying Multimedia: A Guide for Multimedia Developers from Apple Computer Inc.* Apple Computer Inc, 1993.

While this book, which is directed primarily at managers, has negligible technical content and is dated (it is mostly about CD-ROM production), it is good on the process of multimedia production — especially on the different jobs involved — and remains worth reading. It includes interviews with people working in multimedia, and case studies.

[App93b] Apple Computer, Inc. *Inside Macintosh: Text.* Addison-Wesley, 1993.

Most of this book is the reference manual for the (soon to be obsolete) MacOS text APIs, and therefore is of limited interest. However, the introduction contains one of the best available treatments of scripts and writing systems, and the difficulties they present for displaying text on computers for an international audience. Worth looking at, therefore, if you come across a copy.

[Bar77] Roland Barthes. *Image - Music - Text.* Fontana, 1977.

Nobody who is at all interested in the cultural aspects of media and multimedia can afford to neglect this collection of essays. Barthes has written several other books, and if you find his ideas interesting and wish to pursue them further, you should track them down.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 572
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[Bax94] Gregory A. Baxes. *Digital Image Processing: Principles and Applications.* John Wiley & Sons, 1994.

Most books on image processing (apart from introductions to tricks you can get up to with Photoshop) are highly mathematical and mainly concerned with image analysis and recognition, so this book is especially valuable, as it covers a range of image manipulations in an accessible way. Many examples of the effects of the processes on sample images are included.

[Bla92] Lewis Blackwell. *20th Century Type.* Laurence King Publishing, 1992.

An extremely good history of the important developments in typography this century. Anyone who has any interest in fonts and typography should read this book.

[BLFM98] Tim Berners-Lee, Roy T. Fielding and Larry Masinter. Uniform Resource Identifiers (URI): Generic syntax. RFC 2396, IETF, 1998.

➲ The formal definition of URIs, which includes URLs as a subset.

[Bra96] Scott O. Bradner. The Internet standards process — revision 3. RFC 2026, IETF, 1996.

How a document becomes an Internet standard: this description of the process provides an insight into what it means to be a standard.

[Car97] Rob Carter. *Experimental Typography.* Working With Computer Type 4. RotoVision, 1997.

More of a guide than other books on the subject, which tend to be mere showcases. This book includes a description of the rules traditionally used for layout, and then goes on to offer advice about breaking them. There are plenty of examples.

[CHW99] Jon Crowcroft, Mark Handley and Ian Wakeman. *Internetworking Multimedia.* Taylor & Francis, 1999.

➲ A comprehensive account of the issues we described briefly in Chapter 15, including multicasting, protocols and videoconferencing.

[Cli94] Stafford Cliff. *The Best in Cutting Edge Typography.* Quarto, 1994.

This book captures the grunge typography movement at its peak, with lots of examples including record covers, logos and stationery, posters and advertisements, all showing type being used as a graphic element in new ways, partly because of the possibilities opened up by digital fonts.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 573
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[Con87] Jeff Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 20(9):17–41, September 1987.

An early standard survey reference on hypertext and hypermedia, which has not dated as much as you might expect. Although the World Wide Web has appeared and grown up since the publication of this paper, many of the issues Conklin considers remain relevant, and are only now being addressed with the development of XML.

[CSS] Cascading Style Sheets, level 2, `http://www.w3.org/TR/REC-CSS2`.

➲ This document defines the CSS language, including absolute positioning facilities. Hopefully, by the time you read this, Web browsers will have correctly implemented it.

[DOMa] Document Object Model (DOM) level 1 specification, `http://www.w3.org/TR/REC-DOM-Level-1`.

➲ Unfortunately, this document only defines the basic parts of the DOM, which has left browser manufacturers free to implement the interesting bits as they see fit. See also [DOMb].

[DOMb] Document Object Model (DOM) level 2 specification, `http://www.w3.org/TR/WD-DOM-Level-2`.

➲ Level 2 of the DOM defines the parts of the model dealing with events and stylesheets, i.e. the parts used in 'dynamic HTML'. The standard is not, at the time of writing, much observed, but should be the shape of the DOM in future.

[ECM97] ECMA. *ECMAScript: A general purpose, cross-platform programming language.* Standard ECMA-262, 1997.

➲ The reference manual for the language usually known as JavaScript. Does not describe any host systems, so cannot be used as a complete reference for Web scripting.

[EF99] Elaine England and Andy Finney. *Managing Multimedia: Project Management for Interactive Media.* Addison-Wesley, 2nd edition, 1999.

The complexity of many multimedia projects implies the need for management — a topic we have barely touched on. This book is devoted to it.

[Fau98] Christine Faulkner. *The Essence of Human-Computer Interaction.* Prentice-Hall, 1998.

A good, concise survey of mainstream thinking about HCI and usability.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 574
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[FGM+97] Roy T. Fielding, Jim Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen and Tim Berners-Lee. Hypertext Transfer Protocol — HTTP/1.1. RFC 2068, IETF, 1997.

➲ Full details of the latest version of HTTP, with all the request and response headers, and status codes.

[Fla97] David Flanagan. *JavaScript: The Definitive Guide.* O'Reilly & Associates, Inc., 2nd edition, 1997.

➲ There are many books currently available on JavaScript, JScript, or DHTML, but this is one of the best, providing a clear and thorough account of the JavaScript language and its uses on the World Wide Web. The book does not cover the standards (ECMAScript and the DOM), but it does cover the real implementations as they stood at the time of its publication. Expect a revised edition when the browser manufacturers come into line with the W3C Recommendations.

[FvDFH96] James D. Foley, Andries van Dam, Steven K. Feiner and John F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, 2nd edition, 1996.

➲ This well-established academic text contains a thorough account of the theory of computer graphics. It concentrates largely on vector graphics, and includes an extensive account of all aspects of 3-D graphics. The authors' approach is mainly mathematical, and many readers will find it dry. Available in both Pascal and C editions.

[Got98] Veruschka Gotz. *Digital Media Design: Color and Type for the Screen.* RotoVision, 1998.

An excellent succinct introduction to the issues raised by the differences between display and print in connection with colour and typography. Instead of laying down rules based on established practice, the author encourages us to look at digital media in their own right, and to take advantage of their special qualities.

[GQ99] Ian S. Graham and Liam Quin. *XML Specification Guide.* John Wiley & Sons, 1999.

➲ The title parses as 'a guide to the XML specification': the book is mostly an annotated version of the specification [XML]. For anyone seriously interested in XML, this is essential reading.

[Gra96] Ian S. Graham. *HTML Sourcebook.* John Wiley & Sons, 4th edition, 1998.

One of the better books on HTML. Although there are many others to choose from, this is clear and accurate.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 575
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[Gra97] Ian S. Graham. *HTML Stylesheet Sourcebook.* John Wiley & Sons, 1997.

A good description of CSS. It includes a clear presentation of the arguments in favour of stylesheets, a description of CSS1 and most of CSS2 (although this was only a proposal at the time this book was written), with many examples of the layout and typographic effects that can be achieved. Includes rather too much description of the bugs in browsers' implementations of CSS, which should be out of date by now.

[HF97] Steven Heller and Anne Fink. *Faces on the Edge: Type in the Digital Age.* Van Nostrand Reinhold, 1997.

A collection of experimental typefaces, made using digital technology. The book itself is an example of some of the layout ideas that go with experimental faces.

[HTMa] HTML 4.0 specification, `http://www.w3.org/TR/REC-html40`.

➲ The current HTML specification is based on an SGML DTD, which makes it tough reading if you are not familiar with SGML or XML. This is the standard, and you should try to stick with it, even though many media-rich Web pages rely on non-standard elements, such as `EMBED`.

[HTMb] Timed Interactive Multimedia Extensions for HTML (HTML+TIME), `http://www.w3.org/TR/NOTE-HTMLplusTIME`.

➲ This document only has the status of a 'note', and the proposal has not been adopted in this form by the W^{3}C, but it is worth reading, nevertheless, especially since its sponsors include some heavyweight and influential software manufacturers.

[HW96] Jed Hartman and Josie Wernecke. *The VRML 2.0 Handbook: Building Moving Worlds on the Web.* Addison-Wesley, 1996.

➲ A tutorial and reference on VRML, showing how the language is used to construct models, arrange them into scenes and add interactivity.

[ICC97] ICC profile format specification, version 3.4. International Color Consortium, 1997.

➲ The complete definition of the format of the profiles used by colour management software, such as ColorSync. Much of it is concerned with print issues, but multimedia authors must be increasingly aware of colour management. The specification is presently available in PDF from the ICC's Web site at `http://www.color.org/profiles.html`.

[ISO] Introduction to ISO, `http://www.iso.ch/infoe/intro.htm`.

ISO's own account of their history and rôle in the standards process. The page includes a good discussion of the nature of standards.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 576
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[JMF94] Richard Jackson, Lindsay MacDonald and Ken Freeman. *Computer Generated Color: A Practical Guide to Presentation and Display.* John Wiley & Sons, 1994.

A short introduction to the perception, representation and display of colour, covering physics, psychology and physiology, as well as computational topics. The book is arranged as a series of short articles, which gives it a superficial flavour, reminiscent of popular science magazines, but it actually contains much useful information on a topic that is generally somewhat neglected.

[Kaw92] Bruce F. Kawin. *How Movies Work.* University of California Press, 1992.

Film shares many qualities with multimedia, but it is a more mature technology, with well-developed conventions and working practices. Understanding film can provide useful insights into multimedia, and possibly show us some ways in which it might develop. This book is a comprehensive introduction to all aspects of film. (The author is a professor of English and film studies, so his account of the importance of critics should be taken with a pinch of salt.)

[Kos98] Dave Kosiur. *IP Multicasting: The Complete Guide To Interactive Corporate Networks.* John Wiley & Sons, 1998.

➲ Despite the intimidating sub-title, this is actually one of the most readable books on networking that we know. It clearly describes the motivation behind multicasting as well as the technology itself, including the main protocols used for multimedia.

[Lay98] Kit Laybourne. *The Animation Book.* Three Rivers Press, 'new digital' edition, 1998.

An excellent practical introduction to the techniques and equipment used in animation. As the 'new digital' tag indicates, the current edition takes account of the impact of computers on the animator's craft, but it still includes an extensive coverage of traditional media, and the author is at pains to emphasize the common underpinnings of both approaches.

[LH99] David Lowe and Wendy Hall. *Hypermedia and the Web: An Engineering Approach.* John Wiley & Sons, 1999.

The authors of this book believe that the application of software engineering principles to multimedia production will improve quality and reduce costs. This seems to be a somewhat dubious claim from several points of view, but if you share their faith, you may find this book helpful.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 577
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[Lis95] Martin Lister, editor. *The Photographic Image in Digital Culture.* Routledge, 1995.

A collection of analytical essays, which are relevant inasmuch as the 'photographic image' is broadly defined to include video, and some of the contributors consider interactivity, too.

[McC96] Malcolm McCullough. *Abstracting Craft: The Practiced Digital Hand.* MIT Press, 1996.

The expression 'thought-provoking' is overworked, but it is hard to describe this book any other way. The author raises many questions about the relationship between traditional craft and digital technology, and, indeed, between people and digital technology. His answers may not always be convincing, but the issues are of considerable importance to anybody working in multimedia. If you only read one book from this list, this should be it.

[MS99] Daniel Minoli and Andrew Schmidt. *Internet Architectures.* John Wiley & Sons, 1999.

A classic example of network writing, full of acronyms, neologisms, and poorly constructed sentences, this book is hard going, but contains valuable information about where the Internet is heading.

[Mv96] James D. Murray and William vanRyper. *Encyclopedia of Graphics File Formats.* O'Reilly & Associates, Inc., 2nd edition, 1996.

The bulk of this book comprises descriptions of the internal layout of nearly 100 graphics file formats, including those commonly encountered in multimedia (GIF, JFIF, PNG, etc.). As such, it will only be of interest to programmers. The introductory chapters provide a useful review of some aspects of computer graphics and a general description of how the necessary information can be stored in a file. Extensive pointers to defining documents, standards, and so on are included, and the book has a companion Web site, where updated information and descriptions of new formats are available.

[NG96] Mark Nelson and Jean-Loup Gailly. *The Data Compression Book.* M&T Books, 2nd edition, 1996.

A general account of data compression algorithms, aimed clearly at programmers — all the main algorithms are implemented in C in the text. The emphasis is on practicalities — this book will not satisfy mathematicians. Among the algorithms included are Huffman, Lempel-Ziv in all varieties, and JPEG.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 578
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[Oha93] Thomas A. Ohanian. *Digital Nonlinear Editing.* Focal Press, 1993.

Although quite badly dated, this book provides useful insights into digital editing, from the perspective of film editing in particular.

[O'R98] Michael O'Rourke. *Principles of Three-Dimensional Computer Animation.* W.W. Norton & Company, revised edition, 1998.

A good accessible survey of 3-D techniques.

[Poh95] Ken C. Pohlmann. *Principles of Digital Audio.* McGraw-Hill, 3rd edition, 1995.

➲ Since this is the only serious technical book on digital audio, it's fortunate that it is very good.

[Poy96] Charles A. Poynton. *A Technical Introduction to Digital Video.* John Wiley & Sons, 1996.

➲ More technical than introduction, this is the best book on digital video technologies; it also provides the analogue background in some detail. Sadly, it is now somewhat out of date. A second edition, due to be published early in 2001, will be called 'Digital Video and HDTV: Pixels, Pictures and Perception'. If it lives up to the promise of the existing edition, it should be an essential reference for all aspects of digital video.

[RM94] Francis Rumsey and Tim McCormick. *Sound and Recording: An Introduction.* Focal Press, 2nd edition, 1994.

⇨ A practical introduction to the techniques of sound recording — what goes on before digitization.

[RM96] Paul Resnick and James Miller. Pics: Internet access controls without censorship. *Communications of the ACM*, 39(10):87–93, 1996.

A tutorial introduction to the mechanics of PICS, and the rationale behind it. Also available online at `http://www.w3.org/PICS/iacwcv2.htm`.

[Rot92] Joseph Rothstein. *MIDI: A Comprehensive Introduction.* Oxford University Press, 1992.

The description of MIDI software is out of date in the details, but overall, this is a straightforward, accurate and accessible description of the MIDI protocol and how it is used.

[SACM] Michael Stokes, Matthew Anderson, Srinivasan Chandrasekar and Ricardo Motta. A standard default color space for the internet — sRGB, `http://www.w3.org/Graphics/Color/sRGB`.

➲ The document that defines the controversial sRGB colour space.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 579
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[Sas93] Rosemary Sassoon, editor. *Computers and Typography.* Intellect Books, 1993.

A nice little collection of essays, which concentrates on the relevance of established typographical practice to digital typesetting.

[SCFJ96] Henning Schulzrinne, Stephen L. Casner, Ron Frederick and Van Jacobson. RTP: A transport protocol for real-time applications. RFC 1889, IETF, 1996.

➲ Full details of RTP and RTCP. Note that a slightly revised draft of this document was circulated in 1998, which will form the basis of an Internet standard.

[Sma99] Peter Small. *Lingo Sorcery.* John Wiley & Sons, 2nd edition, 1999.

This distinctly eccentric book should be approached with some caution, especially, perhaps, by non-programmers. The author presents a view of object-oriented programming in the context of Director, but without the perspective of conventional object-oriented programming, the result is ... odd. Nevertheless, he succeeds in showing how Lingo can be used in ways that are not immediately obvious, to produce complex multimedia applications.

[SMI] Synchronized Multimedia Integration Language (SMIL) 1.0 specification, `http://www.w3.org/TR/REC-smil`.

➲ The formal definition of SMIL, as an annotated XML DTD.

[SRL98] Henning Schulzrinne, Anup Rao and Robert Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326, IETF, 1998.

➲ Full details of RTSP, with all the methods, status codes, and so on.

[SWDB98] Sean C. Sullivan, Loren Winzeler, Jeannie Deagen and Deanna Brown. *Programming with the Java Media Framework.* John Wiley & Sons, 1998.

➲ Contains reference material on the Java APIs for playing audio and video, and tutorials on how to use them. Purely for Java programmers.

[TDT96] Ronald C. Turner, Timothy A. Douglass and Audrey J. Turner. *README.1ST: SGML for Writers and Editors.* Prentice-Hall, 1996.

⇨ An accessible introduction to SGML, written, as the sub-title indicates, from the perspective of writers and editors. This perspective helps clarify the arguments in favour of structural markup.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 580
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[Tho93] Roy Thompson. *Grammar of the Edit.* Focal Press, 1993.

The title may strike you as strange, if you are not used to thinking about structures in grammatical terms, but you should not be put off. This book is simply an account of different types of shots and transitions used in film and video, how they may be combined, and the effects that result. Useful both as a guide for video editing and as a reference point when considering ways of combining media, especially in time-based multimedia.

[Tud95] P. N. Tudor. MPEG-2 video compression. *Electronics and Communication Engineering Journal*, December 1995.

A very clear short description of MPEG-2 (and, by implication, MPEG-1) compression. Also available (at the time of writing) online at `http://www.bbc.co.uk/rd/pubs/papers/paper_14/paper_14.htm`.

[Vau98] Tay Vaughan. *Multimedia: Making It Work.* Osborne McGraw-Hill, 4th edition, 1998.

Vaughan covers much of the same ground as *Digital Multimedia*, but much more superficially. He does, however, deal with issues such as project management, contracts, and so on, which we have not considered. His style, which is informal and anecdotal, may not be to everybody's taste.

[VD96] Mark Cotta Vaz and Patricia Rose Duignan. *Industrial Light and Magic: Into the Digital Realm.* A Del Rey book published by Ballantine Books, 1996.

⇨ Essentially just an entertaining public relations exercise for ILM, but one that provides an insight into the extraordinary effort (and the technology) that goes into high-end digital special effects. It usefully puts desktop and multimedia post-production efforts into perspective.

[Vin92] John Vince. *3-D Computer Animation.* Addison-Wesley, 1992.

An unpretentious introduction to 3-D modelling, rendering and animation techniques.

[WAI] WAI accessibility guidelines: Page authoring, `http://www.w3.org/TR/WD-WAI-PAGEAUTH`.

Compulsory reading for anybody designing Web pages.

[Wil74] Raymond Williams. *Television: Technology and Cultural Form.* Fontana, 1974.

⇨ An influential study of the cultural aspects of television, which provides an important reference point for any attempts to analyse the cultural impact of multimedia.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 581
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

[Wil93] Adrian Wilson. *The Design of Books.* Chronicle Books, 1993.

⇨ A description of the practice and values of traditional book design, from a leading designer. Pre-dates the digital revolution, but contains many things that remain relevant.

[WW92] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques: Theory and Practice.* ACM Press/Addison-Wesley, 1992.

➲ If you want to know how ray tracing, radiosity, and other advanced 3-D algorithms work, this book will tell you, in considerable technical detail.

[XML] Extensible Markup Language (XML) 1.0, `http://www.w3.org/TR/REC-xml`.

➲ The official XML specification is dense and quite hard to read; generally, you are better off with [GQ99]. Related specifications (XPointer, XLink, XSL, and others) can be traced from `http://www.w3.org/XML`.

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 582
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

Index

3-2 pulldown, 307
3-D animation, 367–370
3-D graphics, 104–119

A-law, 397
absolute positioning, 257
abstract characters, 186, 206
access to multimedia, 17–29
  and disabilities, 19–20, 442–445
  and PDAs, 46
Acrobat
  Distiller, 266
  Exchange, 266
    adding links in, 276
  Reader, 265
  see also PDF
actions, 447, 472
Adaptive Differential Pulse Code Modulation, *see* ADPCM
Adobe Systems, Inc., *see* After Effects, Ilustrator, PDF, Photoshop, PostScript, Premiere
Adobe Type Manager, 212
ADPCM, 398
ADSL, 55
After Effects, 337, 363
  interpolation in, 364
  value graphs, 364
AIFF, 402
aliasing, 43, 385
  see also anti-aliasing
alignment of text, 254
  in CSS, 255
alpha channels, 137–138, 177
alphabets, 187
analogue representations, 37
  and noise, 39, 304
  reconstruction of, 40, 43
anchor points, 98
anchors, 281
  images as, 423
  in SMIL, 439
animated GIFs, 354–355, 460
animation, 301, 348–373
  3-D, *see* 3-D animation
  cel, 349
  of layers in Web pages, 464, 471
  sprite, 357
  stop-motion, 350
  use of scripts for, 460–462
anti-aliasing, 90–91, 137
  of fonts, 214
applets, 420, 485–489
  embedding in Web pages, 485
  passing parameters to, 485
  security of, 487
ascenders, 208
ASCII, 188–190, 195
aspect ratio, 309
Asymmetric Digital Subscriber Line, *see* ADSL
ATM networks, 515
attributes, 238
AU, 402
audio, *see* sound
authoring systems, 9, 50
  for HTML, 468
    see also Director, Dreamweaver
AV disks, 49
AVI, 324, 325
axes, 87

B-pictures, 320
backward prediction, 320
bandwidth, 340, 396, 435, 514
base URL, 280
baseline, 208, 252
behaviours, 448, 466–472

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 583
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

applied to objects in 3-D, 368
in PDF, 467
Bézier curves, 92–98
combining, 96
control points, 93
direction points, 93
drawing, 93, 97
bitmapped fonts, 211
scaling, 211
bitmapped images, 69, 121–151
combining with vector graphics, 74–76
compared with vector graphics, 70–74
embedding in HTML, 420
geometrical transformation of, 148
using in drawing programs, 76
bits, 35
blue screen, 177, 336
blurring, 143, 145
BMP, 82, 163
bold fonts, 202, 213, 251
bookmarks, 292
brightness, 169
brightness and contrast adjustments, 138
Bryce, 117
animation in, 370
bump mapping, 116
bytes, 35, 158

caching of Web pages, 507–509
calligraphic fonts, 201
Canoma, 119
cap height, 208
Cascading Style Sheets, *see* CSS
CCIR 601, 309
CD-ROM, 6
censorship, 25–28
CGI, 58, 279, 516–519
limitations of, 520
query strings, 517
channels, 175
character entity references, 236
character repertoire, 187
character sets, 36, 187–196
ASCII, 188–190
in URLs, 278
see also ISO 646
encodings, 194
Quoted-Printable, 194
UCS-2, 195
UCS-4, 195
UTF-16, 193
see also UCS Transformation Formats
in HTML, 241
ISO 10646, 192–196
and HTML, 235
Basic Multilingual Plane (BMP), 193
see also Unicode
ISO 646, 189–190
ISO 8859, 191
platform-specific, 190
chroma downsampling, 177
chroma keying, 336
chromaticities, 178
chrominance, 308
chrominance sub-sampling, 310, 321
CIE chromaticity diagram, 157
CIF, 344
Cinepak, 317
CJK consolidation, 192
class attribute (HTML), 240, 246, 247
client-side scripts, 450
restrictions on, 451
clients, 57, 520
World Wide Web, 57
clipping, 390
CLUT, *see* palettes
code points, 187
code values, 187
codecs, 304, 317–322, 324
animation, 356
asymmetrical, 314, 317
for streamed video, 343–344
symmetrical, 314
ColorSync, 181, 553
colour
16 bit, 159
24 bit, 158
additive mixing, 158
and CSS, 254
and the World Wide Web, 253
CMYK, 166–169
device-independent specification of, 174
direct, 160
indexed, 160–166
physics of, 154
RGB, 156–166
saturated, 170
sRGB, 182, 254
subtractive mixing, 158, 167
colour correction, 175–176
colour depth, 158, 338
and image file size, 160
colour gamut, 156
CMYK, 168
RGB, 156
colour management, 179
colour pickers, 172
colour profiles, 179
ICC, 181
colour temperature, 178
colour vision, 155
colour wheel, 170
comments
CSS, 243
ECMAScript, 456
HTML, 237
Common Gateway Interface, *see* CGI
companding, 397
complementary colours, 167
compositing, 75, 78, 141
video, 336
with alpha channels, 137
compression
dictionary-based, 128
inter-frame, 312
intra-frame, 312
Lempel-Ziv, 128
lossless, 80, 127–128
lossy, 80, 127, 397, 399
see also JPEG
LZW, 128
of images, 80, 125–133
of sound, 396–402
perceptually based, 399–402
of speech, 397–399
of video, 302, 312–322
see also codecs, MPEG
spatial, 312
temporal, 312, 319

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 584
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

compression artefacts, 133, 301
condensed fonts, 202
constructive solid geometry, 107
control characters, 189
convolution, 142-147
cookies, 520
coordinate geometry, 87-89, 104
  use of equations in, 89
coordinate transformations, 88
coordinates, 87, 465
CSS, 239-244, 249-262
  absolute positioning, 257
    and scripts, 464
  alignment of text in, 255
  and fonts, 249-253
  and images, 421
  colour and, 254
  declarations, 240
  inheritance, 251
  properties, 243, 249-262
    changing with scripts, 463
  pseudo-classes, 282
  rules, 239
  selectors, 240, 249

DAT, 384, 389
data rate
  of CCIR 601 video, 310
  of CD-quality sound, 389
  of compressed sound, 401
  of DV, 316
  of hard disks, 48
  of motion JPEG, 315
  of MPEG-2, 311
  of video, 302
  see also bandwidth
data structures, 36
datagrams, 495
DC component, 42
DCT, 129-130, 142, 319, 344
  inverse, 131
de-interlacing, 307
definition of multimedia, 12
delays, 513
descenders, 208
difference frames, 313
digital audio tape, *see* DAT
Digital Versatile Disk, *see* DVD
Digital Video Disk, *see* DVD
digitization, 37-45
  of sound, 383-388
  of video, 302-305
dingbats, 193, 207
Director, 10, 52, 425
  and the World Wide Web, 493
  behaviours, 479
  cast, 425
  Lingo, *see* Lingo
  movies, 425
  object model, 480
  score, 425
  scripting, 479
  Xtras, 483-484
Discrete Cosine Transform, *see* DCT
display fonts, 203
dithering
  of colours, 164
  of sound signals, 386
document elements, 228, 235
  block-level, 244
  classes of, 240
  DIV, 246
  empty, 238
  inline, 244
  parent, 235
  SPAN, 247
Document Type Definition, *see* DTD
DOM, 451
  level 2, 463
domain names, 277
dominant wavelength, 169
dots per inch, 122
downsampling, 124
drawing programs, 71, 73
  rulers in, 87
  see also Illustrator
Dreamweaver, 468, 478
  adding behaviours to, 471
  behaviours palette, 468
  timeline, 471
    and scripts, 477-479
DTD, 228
  HTML, 241
DV, 303, 311, 331
  compression of, 316
DVD, 7, 311, 315, 322
dynamic HTML, 452
dynamic range, 381

easing, 361
ECMAScript, 451, 453-466, 552
  built-in objects, 453
  comments, 456
  host objects, 454
  methods, 454
  objects in, 454-458
  primitive types, 453
  properties, 454
  see also JavaScript
ellipses
  drawing, 92
em, 209
em-dashes, 209
en, 209
en-dashes, 209
envelope shaping, 395
EPS, 83
ethics, 17, 25
Euro currency symbol, 191, 207
event handlers, 458-462
  in Lingo, 480
event-driven systems, 447
events, 447
ex, 209
extended fonts, 202
eXtensible Markup Language, *see* XML
extrusion, 109

faders, 395
fields, 306
file formats
  for graphics, 80-84
    conversion between, 84
  for sound, 402-403
  MIDI, 405
  MOD, 411
File Transfer Protocol, *see* FTP
fills, 98, 99
  gradient, 100
  pattern, 100
film, 2, 10
  censorship of, 28
  editing, 327-328
    conventions, 325
  frame rate, 300
  history of, 8-9
filters, 42, 103, 142

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 585
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

high pass, 393
low pass, 393
notch, 393
sound, 392
see also pixel group processing
firewalls, 509
FireWire, 48, 50, 303, 331
Flash, 75, 359, 362, 537
and HTTP requests, 519
behaviours in, 472–476
buttons, 472
easing in, 361
interpolation in, 361
stage, 360
symbols, 360
timeline, 360
using bitmaps in, 360
vectorizing in, 75
font families, 203
font metrics, 210
fonts, 196–216
and CSS, 249–253
bitmapped, 211
scaling, 211
body size, 208
bold, 202, 213, 251
calligraphic, 201
condensed, 202
dingbat, 207
display, 203
embedding in files, 198
extended, 202
italic, 201
modern, 210
monospaced, 199
multiple master, 214–216
design axes, 214
instances, 215
naming of, 250
old-style, 210
OpenType, 213
outline, 211
scaling, 212
pi, 206
proportional, 199
Roman, 200
sans serif, 200
serifed, 200
slanted, 201
small caps, 201
substitution of, 199, 206
and multiple master fonts, 215
in PDF, 216
symbol, 206
text, 203
TrueType, 212
instructions, 214
Type 1, 212
hints, 213
upright, 201
weight of, 202
forensic multimedia, 3
Fourier Transform, 41, 129
fractals, 109
random, 111
fragment identifiers, 281
frame grabbing, 351
frame rate, 338
frame stacks, 353
framesets, 247
and links, 284
free form modelling, 108
frequency, 41
negative, 42
of sound, 378
frequency domain, 41, 90, 129
frequency spectrum, 41, 378
FTP, 57
fusion frequency, 300

games, 3
interactivity in, 14
gamma, 179, 339
Gaussian blur, 144, 146
General MIDI, 407
GIF, 81
and indexed colour, 163
animated, 354–355
Gimp, The, 134
glyphs, 196, 206
bounding boxes of, 210
Gouraud shading, 114
gradient fills, 100
graphic equalization, 395
graphics
in multimedia, 66
motion, *see* motion graphics
text as, 220–221
graphics metafiles, 83
greyscale, 159
Group of Pictures, 320

H.261, 343
H.263, 344
hard synchronization, 434
harmonics, 41
helper applications, 418
hidden surface removal, 113
hosts, 495
href attribute
HTML, 280, 281, 283
SMIL, 436
HSV colour model, 171
HTML, 57, 198, 231, 234–239, 244–249
and hypermedia, 418–424
class attribute, 240, 246, 247
dynamic, 452
embedded objects, 420–423
embedding scripts in, 455
features to aid accessibility, 443
forms, 517
frames, 247
headers, 244
href attribute, 280, 281, 283, 418
id attribute, 248, 283
links in, 277–286
lists, 244
proprietary extensions to, 231
tables, 244
text layout algorithm, 256
HTML+Time, 440–442
HTTP, 57, 194, 277, 281, 418, 494, 503–509
caching, 507–509
requests, 504–506
for CGI scripts, 517
conditional, 508
responses, 506–507
constructed by CGI scripts, 517
status codes, 506
use of MIME types in, 59
HTTP streaming, 341
hue, 169
Huffman coding, 128
of sound, 396
use in JPEG compression, 131
Hypercard, 271

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 586
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

hypermedia, 57, 270, 415, 417–424
see also hypertext
hypertext, 269–297
contextual structures for, 296
history of, 270–272
see also links
HyperText Markup Language, *see* HTML
HyperText Transfer Protocol, *see* HTTP

I-pictures, 320
IAB, 62
IANA, 59, 62, 194, 495
id attribute
HTML, 248, 283
SMIL, 431
IEC, 61
IEEE 1394, *see* FireWire
Illustrator, 71, 92, 94
autotrace tool, 75
brush strokes in, 76
files, 84
filters in, 103
gradient mesh tool, 100
pen tool, 93
pencil tool, 98
PostScript generated by, 82
image compression, *see* compression of images
image histograms, 139
image manipulation, 133–151
applied to video, 335
image maps, 423–424, 439
image resolution, 123
in points, 328, 332
in-betweening, 359
indexing of Web sites, 293–296
inheritance
in CSS, 251
in Lingo, 482
Integrated Services Architecture, 515
Integrated Services Digital Network, *see* ISDN
Intel Indeo, 317
interactivity, 3, 13–17, 416, 448
interlacing, 306
interleaving, 412
International Organization for Standardization, *see* ISO
International Telecommunication Union, *see* ITU
Internet, 53, 493
demands on, 500
dial-up connections to, 54
quality of service and, 515
Service Providers, *see* ISPs
standards and, 62
see also World Wide Web, TCP/IP
Internet Architecture Board, *see* IAB
interpolation, 72, 124, 149–151, 359, 364
bi-cubic, 150
bi-linear, 150
nearest neighbour, 150
intranets, 58, 493
inverse kinematics, 369
IP, 495–497
addresses, 495
for multicasting, 501
ISDN, 54, 343
ISO, 60, 61
ISO Latin1, 191, 192, 195
ISPs, 53, 58
and quality of service, 515
Web caching by, 509
italic correction, 263
italic fonts, 201
ITU, 61, 156

jaggies, 90
see also aliasing
Java, 485, 520
Media Framework, 448, 489, 521
virtual machine, 486
JavaScript, 451, 551
see also ECMAScript
JFIF, 81
jitter, 341, 385, 514
JPEG, 81, 129–133, 163
motion, *see* motion JPEG
quality setting, 132
quantization matrix, 131, 132
zig-zag sequence, 131
JPEG2000, 132
JScript, 451

kerning, 210
key frames, 335, 358
and video compression, 313
attaching actions to, 472
in Director, 426
in Flash, 359
interpolation between, 359
keying, 336
kinematics, 369
inverse, 369

LANs, 53, 56, 58
and multicasting, 501
see also intranets
lathing, 109
layers, 77–80
adjustment, 80
animating with, 356, 363
combining, 78
tracing with, 78
left side bearing, 210
letter spacing, 211
levels adjustments, 138
ligatures, 211
lighting, 106, 114
Linear Predictive Coding, 399
Lingo, 479–482
extending, 484
inheritance in, 482
Web extensions to, 493
link types, 285
links, 269, 272, 274–290
between entire documents, 286
bi-directional, 275
destination, 275, 276, 283
display of, 276, 291
images and, 423
in HTML, 277–286
in PDF, 275–276
in SMIL, 436–440
in XML, 287–290
inline, 290
out-of-line, 290
regional, 275
semantics of, 285
simple uni-directional, 275
source, 275, 276, 281
Liquid Audio, 404

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 587
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

LiveScript, 451
local area networks, *see* LANs
localization, 226
luma keying, 336
luminance, 157, 173, 308

macroblocks, 319
Macromedia, Inc., *see* Director, Dreamweaver, Flash
markup, 222
  structural, 224–227, 232, 444
    weaknesses of, 262–263
  visual, 224
markup languages, 219, 426
masking curve, 400
masking of sounds, 400
masks, 136
  greyscale, 137
mattes, 336
  travelling, 337, 365
Media Cleaner Pro, 340
media time, 433
metaballs, 111
MetaCreations Corp., *see* Bryce, Painter, Poser
metadata, 295
MIDI, 404–411
  controllers, 406
  editing, 409
  messages, 406–407
MIME types, 58–60, 194, 418, 421, 505
  of stylesheets, 243
  of scripts, 456
MJPEG, *see* motion JPEG
modelling
  3-D, 107–112, 116–119
  figures, 117–118
  free form, 108
  hierarchical, 106
  image-assisted, 119
  physics-based, 111
  procedural, 109–111
  terrain, 110, 117
modems, 54
  cable, 55
modern fonts, 210
monospaced fonts, 199
monostyled text, 198
motion compensation, 316, 319
motion graphics, 363–366, 426, 470
motion JPEG, 314–316
motion paths, 361, 367
movie controllers, 323
MP3, 391, 401, 403
  streamed, 403
MPEG, 318–322, 324
  audio, *see* MP3
MPEG-2, 311
  profiles, 311
MPEG-4, 318, 416
μ-law, 397
multicasting, 500–502, 515
  and LANs, 501
  host groups, 501
    management of, 501
multimedia applications, 11
  distributed, 519
multimedia architectures, 52
multimedia in business, 4
multimedia in education, 4
multimedia PC specifications, 45
multimedia presentations, 416
multimedia productions, 11
multiple master fonts
  see under fonts
multiple media, 12
Musical Instruments Digital Interface, *see* MIDI

network computers, 489
nodes, 274
noise gates, 393
non-linearity, 272–274
non-zero winding number rule, 101
NTSC, 305
  CCIR 601 sampled, 309
  frame rate, 300, 306
  resolution, 122
numbered image sequences, 352
numeric character references, 236
NURBs, 108
Nyquist rate, 42

objects
  in ECMAScript, 454–458
  in Lingo, 481
  media, 428
offline delivery, 6
old-style fonts, 210
onion skinning, 353
online delivery, 6, 7, 493
online stores, 24, 507
OpenType fonts, 213
optical character recognition, 197
out points, 328, 332
outline fonts, 211
  scaling, 212
oversampling, 124
overshoot, 213

P-pictures, 320
packet-switched networks, 494
packets, 494
page layout, 420
Painter, 71
  animation in, 352
  colour mixing in, 169
  frame stacks, 353
painting programs, 67, 73
  see also Painter, Photoshop
PAL, 305
  CCIR 601 sampled, 309
  frame rate, 301
  resolution, 122
palettes, 161, 162
  Web-safe, 165
particle systems, 111
paths, 97
  closed, 97
  motion, 361
  open, 97
pattern fills, 100
PDF, 264–266
  and behaviours, 467
  font substitution in, 216
  links in, 275–276
persistence of vision, 300, 306, 348
personal digital assistants, 46
Phong shading, 114
Photoshop, 10, 71, 182, 363, 550
  curves dialogue, 140
  files, 84
  Gaussian blur, 144
  interpolation in, 149
  lasso tool, 135

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 588
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

layers, 77
adjustment, 80
magic wand, 135
magnetic lasso, 136
marquee tools, 135
plug-ins, 134, 335
vector-based text in, 221
pi fonts, 206
pica (pc), 207
PICS, 27-29, 295
pitch alteration, 395
pixel dimensions, 122
pixel group processing, 141-148
pixel point processing, 138-141
pixels, 68
CCIR 601, 310
identifying by coordinates, 87
logical, 69
physical, 69
pixels per inch, 123
Platform for Internet Content Selection, *see* PICS
plug-ins
Photoshop, 134, 335
Shockwave, 493
Web browser, 53, 419
PNG, 81, 128, 133, 159, 163
points (pt), 207
polygon meshes, 108
polylines, 92
port numbers, 498
Portable Document Format, *see* PDF
Poser, 117-118, 369
post-production, 335-338
posterization, 44, 164
PostScript, 70, 82, 120, 264, 549
and SVG, 83
encapsulated, *see* EPS
fonts, 212
line caps in, 99
line joins in, 99
points, 208
Premiere, 331
monitor window, 332
project window, 331
sound effects plug-ins, 393
sound in, 411
stop frame capture in, 351
timeline window, 331
transitions in, 334
presentation time, 433
primary colours
additive, 155
subtractive, 168
procedural modelling, 109-111
process colours, 168
progressive scanning, 306
proportional fonts, 199
protocols, 57, 494-500
in URLs, 277
layered, 494
see also entries for individual protocols

QCIF, 344
quality of service, 514
quantization, 39, 385-388
of rhythm, 44, 409
quantization levels, 39, 385
insufficient, 44
non-linear, 397
quantization noise, 44, 385
QuickTime, 24, 59, 84, 322-325, 413, 434
and animation, 355
and AVI, 324
and file format conversion, 84
and MIDI, 407
and SMIL, 427
API, 489
as a multimedia architecture, 52
audio CD import, 391
components, 323
file format, 324
for Java, 324
MIME type for, 60
MJPEG-A format, 315
movies, 323
self-contained, 339
Musical Instruments, 407
plug-in, 324, 373, 419
parameters for, 422
sound in, 403
sprite tracks, 357
streaming, *see* Streaming QuickTime
VR, 373

radiosity, 115
RAID arrays, 48
rasterizing, 75
ray tracing, 114, 117, 368
RealAudio, 403
RealVideo, 342
codec, 344
rectangles
drawing, 92
reflection, 102
rendering, 69, 89
3-D, 105, 106, 112-116
resolution, 73, 121-125
image, 123
reverb, 394
RLE, 126, 356
use in JPEG compression, 131
robots, 294
rollover buttons, 459, 470
Roman fonts, 200
rotation, 102
in 3-D, 105
rotoscoping, 354
routers, 496
and multicasting, 501
RSAC ratings, 29
RTCP, 500
RTP, 343, 499-500, 510
payloads, 499
profiles, 500
RTSP, 343, 510-513
presentation descriptions, 511
session identifiers, 511
run-length encoding, *see* RLE

S-video, 308
samplers, 407
sampling, 39, 90, 122, 149
4:1:1, 310
4:2:0, 310
4:2:2, 310
of sound, 384
of video, 309
sampling rate, 39, 384, 389
inadequate, 40
Sampling Theorem, 42
sans serif fonts, 200
saturation, 169
scaling, 102

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 589
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

of bitmapped images, 72, 148
of vector graphics, 73
scene-based multimedia, 416
scripting languages, 448–451
scripts
and stylesheets, 462–466
CGI, 279, 516–519
client-side, 450
restrictions on, 451
Director, 479–482
embedding in HTML, 455
Painter, 353
server-side, 450, 516
use of for animation, 460–462
search engines, 24
SECAM, 305
selections, 135–136
feathered, 137
sequencers, 405
software, 409
serifed fonts, 200
serifs, 200, 210
server-side scripts, 450
servers, 56, 492, 503, 520
HTTP, 516
and MIME types, 418
World Wide Web, 57, 494
SETI@home, 30
SGML, 227–231, 233
shading algorithms, 114
sharpening, 145
shearing, 102
applied to fonts, 201
Shockwave, 53, 493
Flash, *see* SWF
signals, 38
slanted fonts, 201
small caps fonts, 201
SMIL, 233, 426–440, 512
Basic Layout Language, 427
clock values, 429
event values, 431
linking element types in, 436
regions, 427
synchronization elements, 428
soft synchronization, 434
Sorenson video codec, 317
sound, 376–413
digitization of, 383–388
editing, 391–392
effects, 394–396
importing from CD, 391
loops, 392
physics of, 377–379
psychological aspects of, 382–383
recording, 388–391
setting levels, 390
sampling of, 384
streamed, 403–404
Source Input Format, 321
spectral power distributions, 154
specular reflection, 114
spiders, 294
SPIFF, 81
sprites, 357
in Director, 425
sRGB colour model, 182, 254
Standard Generalized Markup Language, *see* SGML
standards, 60–63
for character sets, 188–196
for video, 305–312
international, 61
Internet, 62
MPEG, 318
stereophony, 382
storyboards, 10
Streaming QuickTime, 342
and RTSP, 511
recommended codec, 344
strokes, 98
dashed, 99
line cap, 99
stylesheet languages, 229
stylesheets, 229, 239–244, 249–262
and accessibility, 444
and anchors, 282
and scripts, 462–466
links to, 259
see also CSS
SVG, 83, 86
SWF, 83, 86, 362
sound in, 403
symbol fonts, 206
synchronization
hard, 434
in SMIL, 427–434
loss of, 514
of RTP streams, 500
of sound and video, 411–413
soft, 434
synchronization-based multimedia, 415, 425–442
Synchronized Multimedia Integration Language, *see* SMIL
synthesizers, 407

tags, 223, 235, 236
SGML, 228
TCP, 497–499, 510
TCP/IP, 53, 493
and LANs, 58
text fonts, 203
text formatting
tag-based, 222
WYSIWYG, 222
texture mapping, 115
TGA, 82, 163
threshold of hearing, 399
TIFF, 82, 163
time bases, 323
time stretching, 395
timecode
and sound, 411
drop frame, 329
SMPTE, 329
timelines, 415
for multimedia authoring, 425
for sound editing, 392
in Flash, 360
in Premiere, 331
tracking, 211
tracks, 332, 392
transitions, 325, 328
translation, 102
transport addresses, 498
tremolo effect, 395
tristimulus theory of colour, 155
TrueType fonts, 212
instructions, 214
typefaces, 203
see also fonts
Type 1 fonts, 212
hints, 213
typography, 185
and CSS, 249

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 590
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043

grunge, 204
specialized terms in, 207-211

UCS Transformation Formats, 195
UDP, 499
undersampling, 40, 43
unicasting, 501
Unicode, 192, 195, 207, 213
support in ECMAScript, 453
Uniform Resource Locators, *see* URLs
unsharp masking, 146
upright fonts, 201
URIs, 277
URLs, 57, 248, 277-279, 292, 503
base, 280
escape sequences in, 278
fragment identifiers, 281
in RTSP requests, 510
partial, 279
path component of, 277
query strings appended to, 517
use in SMIL, 429
user agents, 235
non-visual, 443
user interfaces, 16-17, 483
event-driven, 447
for time-based media, 16
innovative, 16
use of icons in , 68
UTF-16, *see* UCS Transformation Formats
UTF-8, *see* UCS Transformation Formats

vector graphics, 69, 86-120
3-D, *see* 3-D graphics
animating, 359
combining with bitmapped images, 74-76
compared with bitmapped images, 70-74
origin of term, 88
transformations, 72, 101-103
vector quantization, 317
vectorizing, 75
vectors, 88
3-D, 105
video, 300-344
capture cards, 303
component signals, 308
composite signals, 304, 308
data size, 302
digitization of, 302-305
editing, 325-334
conventions, 325
digital, 330-334
traditional, 328-330
interactive, 342
live, 341
post-production, *see* post-production, 325
progressive download, 341
resolution, 122
streamed, 340-344
see also NTSC, PAL
video conferencing, 56, 342, 492
Video for Windows, 325
virtual reality, 371-373
Virtual Reality Modeling Language, *see* VRML
visualization, 5, 67, 69, 104
voices, 407
voxels, 86
VRAM, 160
VRML, 371-372, 416

W3C, 63
accessibility guidelines, 442
Document Object Model, *see* DOM
WAV, 402
waveforms, 379, 392
use for synchronization, 412
Web, *see* World Wide Web
Web Accessibility Initiative, 444
Web browsers, 269, 510, 517
and distributed multimedia applications, 519
as host environments, 454
downsampling in, 125
executing applets in, 487
history list, 291
plug-ins for, 53, 419
Web portals, 24, 296
Web proxies, 509
Web rings, 297
Web sites, 279
white point, 178
wire frame previews, 113
World Wide Web, 6, 8, 33, 51, 57, 271, 415
and sRGB colour, 182
as a graphical medium, 68
as publishing medium, 21, 24
clients, 57
colour and, 253
Consortium, *see* W3C
graphic bias of, 19
hypertext links in, 281
index sites, 293
popular sites, 23
QuickTime and, 324
scripting and, 450
searching, 292
servers, 57
standards, 63
video delivery over, 341

x-height, 208
XLink, 287, 289-290
XML, 83, 231-234, 275, 427
links in, 287-290
XPointer, 287-289

$Y'C_BC_R$ colour, 173, 177, 310
YUV colour, 173, 308

DISH TECHNOLOGIES L.L.C., Exhibit 2008, p. 591
Aylo Freesites Ltd. v. Dish Technologies L.L.C., IPR2024-00043