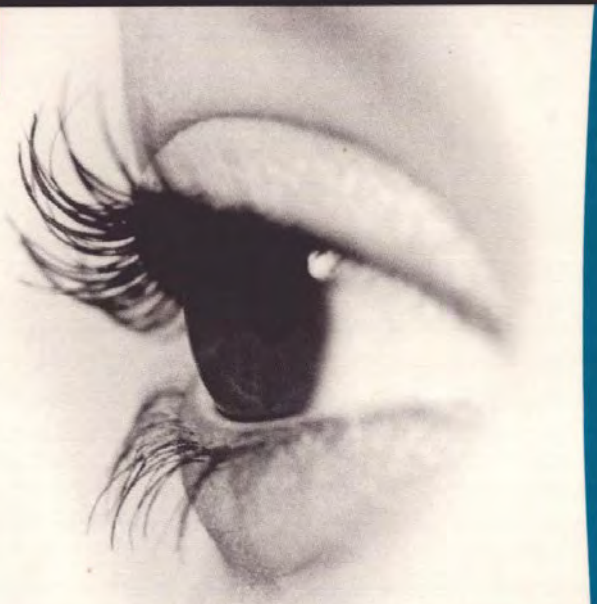
APPENDIX
HARRISON01

WILEY

Symbian OS C++ for Mobile Phones

Volume 3



Application Development for Symbian OS v9

symbian

Richard Harrison
& Mark Shackman

FIFTH THIRD EX1062
PART 2 of 3
Page 2 of 379

symbian

Symbian OS C++ for Mobile Phones

Harrison
Shackman

TK6570
.M6
H295
2007

WILEY

Volume 3



NETWORKING/MOBILE PHONES/C++ £45.00/US $75.00/CAN $97.99

Symbian OS C++ for Mobile Phones
Volume 3

Richard Harrison & Mark Shackman

***Symbian OS C++ for Mobile Phones Volume 3* draws on the experience of Symbian's own engineers to provide you with a thorough grounding in writing C++ applications for mobile phones using the latest version of Symbian OS and beyond.**

symbian press

BICENTENNIAL 1807 WILEY 2007 BICENTENNIAL

This latest edition provides you with the tools to become an effective Symbian OS developer and offers a deep understanding of the fundamental principles upon which Symbian OS is based. The text is also complemented with a suite of examples.

Symbian OS C++ for Mobile Phones Volume 3 includes:

- An introduction to Symbian C++, with descriptions of the basic building blocks and usage patterns.
- Examples that are fully compatible with the latest Carbide.c++ Symbian IDEs guide the reader through increasingly complex aspects of Symbian C++.
- The core concepts, resources and programming idioms that you need to create, test and publish a non-trivial GUI (graphical user interface) application.
- An introduction to some of the most significant Symbian OS system services: extensibility and the use of plug-ins and practical descriptions of the communications, multimedia and database services.
- Advice on the available range of graphics APIs and help to ensure that your application code is as device-independent as possible.

As it is targeted at the high-growth area where computers and mobile communications converge, Symbian OS gives you unparalleled opportunities in mass-market, enterprise and system programming.

If you've programmed in C++ at any level, and wish to develop software for mobile phones, this book is essential.



WILEY
wiley.com

For details of available titles visit:
www.symbian.com/books
www.wiley.com

ISBN 978-0-470-06641-6
9 780470 066416

FIFTH THIRD EX1062
PART 2 of 3

ENGINEERING LIBRARY

Symbian OS C++ for Mobile Phones Volume 3

FIFTH THIRD EX1062
PART 2 of 3
Page 5 of 379

Symbian OS C++ for Mobile Phones Volume 3

Richard Harrison, Mark Shackman

With

Adi Rome, Alex Wilbur, Andrew Jordan, Douglas Feather, Ernesto Guisado, Hassan Ali, Ioannis Douros, John Pagonis, Lucian Piros, Mark Cawston, Martin Hardman, Mathew Inwood, Rick Martin, Sanjeet Matharu, Tim Williams, Yang Zhang

Reviewed by

Graeme Duncan, Guanyun Zhang, Ian McDowall, Jehad Al-Ansari, Jonathan Allin, Jo Stichbury, Kostyantyn Lutsenko, Lane Roberts, Lars Kurth, Mark Jacobs, Mark Welsh, Mathias Malmqvist, Matthew O'Donnell, Rahul Singh, Ricky Junday, Robert Palmer, Rosanna Ashworth-Jones, Sorin Basca, Tim Labeeuw, Warren Day, Will Bamberg

Head of Symbian Press

Freddie Gjertsen

Managing Editor

Satu McNabb



John Wiley & Sons, Ltd

Copyright © 2007 Symbian Software Ltd

Published by John Wiley & Sons, Ltd The Atrium, Southern Gate, Chichester,
West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): cs-books@wiley.co.uk
Visit our Home Page on www.wileyeurope.com or www.wiley.com

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London W1T 4LP, UK, without the permission in writing of the Publisher. Requests to the Publisher should be addressed to the Permissions Department, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, or emailed to permreq@wiley.co.uk, or faxed to (+44) 1243 770620.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The Publisher is not associated with any product or vendor mentioned in this book.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 6045 Freemont Blvd, Mississauga, Ontario, L5R 4J3, Canada

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Cataloging-in-Publication Data

Harrison, Richard.
Symbian OS C++ for mobile phones / By Richard Harrison.
p. cm.
Includes bibliographical references and index.
ISBN 0-470-85611-4 (Paper : allk. paper)
1. Cellular telephone systems – Computer programs. 2. Operating systems (Computers) I. Title.
TK6570.M6H295 2003
621.3845'6 – dc21

2003006223

British Library Cataloguing in Publication Data

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

ISBN: 978-0-470-06641-6

Typeset in 10/12pt Optima by Laserwords Private Limited, Chennai, India
Printed and bound in Great Britain by Bell & Bain, Glasgow
This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

Contents

About the Authors xi

Acknowledgements xix

Symbian Press Acknowledgments xxi

About this book xxiii

Glossary xxvii

1 Getting Started 1
- 1.1 Using the Emulator 1
- 1.2 Hello World – Text Version 6
- Summary 15

2 A System Introduction to Symbian OS 17
- 2.1 Symbian OS Basics 17
- 2.2 Framework Basics 30
- 2.3 APIs Covered in this Book 36
- Summary 39

3 Symbian OS C++ 41
- 3.1 Fundamental Data Types 41
- 3.2 Naming Conventions 43
- 3.3 Functions 49
- 3.4 APIs 51
- 3.5 Templates 55
- 3.6 Casting 56

3.7 Classes 57
3.8 Design Patterns 59
Summary 60

4 Objects – Memory Management, Cleanup and Error Handling 61
4.1 Object Creation and Destruction 62
4.2 Class Categories in Symbian OS 66
4.3 Error Handling 70
4.4 The Cleanup Stack 76
4.5 Two-Phase Construction 81
Summary 85

5 Descriptors 87
5.1 Overview 87
5.2 Anatomy of Descriptors 91
5.3 Literals 93
5.4 Stack Descriptors 93
5.5 Pointer Descriptors 95
5.6 Heap Descriptors 99
5.7 Narrow, Wide and Neutral Descriptors 108
5.8 Descriptors and Binary Data 109
5.9 Using Descriptors with Methods 110
5.10 Some Descriptor Operations 121
5.11 Correct Use of Descriptors 128
5.12 Manipulating Descriptors 138
Summary 148

6 Active Objects 151
6.1 The Asynchronous Service 151
6.2 Multitasking and Pre-emption 156
6.3 A More In-depth Look at Active Objects 158
6.4 How It Works 167
6.5 Active Object Priorities 171
6.6 Active Object Cancellation 176
6.7 Starting and Stopping the Scheduler 181
6.8 Understanding a Stray Signal 182
6.9 Other Common Active Object Errors 187
6.10 Implementing State Machines 190
6.11 Long-Running Tasks and Active Objects 194
Summary 200

7 Files and the File System 203
7.1 File-Based Applications 203
7.2 Drives and File Types 204

FIFTH THIRD EX1062
PART 2 of 3
Page 9 of 379

7.3 File System Services 205
7.4 Streams 216
7.5 Stores 224
Summary 239

8 Interprocess Communication Mechanisms 241
8.1 Overview 241
8.2 Client–server IPC 247
8.3 Publish and Subscribe IPC 253
8.4 Message Queue IPC 257
8.5 Which IPC Mechanism Should You Use? 259
Summary 261

9 Platform Security and Publishing Applications 263
9.1 Releasing an Application 263
9.2 How Does Platform Security Work? 264
9.3 How Do I Support Platform Security? 266
9.4 Preparing an Application for Distribution 268
9.5 Overview of Symbian Signed 272
9.5 Installing a SIS File 279
9.6 List of Capabilities 280
Summary 281

10 Debugging and the Emulator 283
10.1 Using the Emulator 283
10.2 Emulator Debugging 289
10.3 Debugging on a Phone 306
10.4 Miscellaneous Tools 308
Summary 309

11 The Application Framework 311
11.1 Symbian OS Application Framework 311
11.2 S60 and UIQ Platform Application Frameworks 313
11.3 A Graphical Hello World 315
Summary 330

12 A Simple Graphical Application 331
12.1 Implementing the Game on S60 333
12.2 Differences for UIQ 3 355
Summary 366

13 Resource Files 367
13.1 Why a Symbian-Specific Resource Compiler? 367
13.2 Source File Syntax 368
13.3 Bitmaps and Icons 372

13.4 Updating the Resource Files 377
13.5 Application Registration Files 378
13.6 Localizable Strings 379
13.7 Multiple Resource Files 382
13.8 Compiling a Resource File 383
13.9 The Content of a Compiled Resource File 385
13.10 Reading Resource Files 388
Summary 389

14 Views and the View Architecture 391
14.1 The View Architecture 392
14.2 The `MCoeView` Interface 397
14.3 Introduction to the Example Application 398
14.4 Creating and Managing the Views 406
14.5 Implementing the `MCoeView` Interface 408
14.6 Command Menus 414
Summary 421

15 Controls 423
15.1 What Is a Control? 423
15.2 Control Types 424
15.3 Control Layout 429
15.4 Handling Key and Pointer Events 432
15.5 Observing a Control 442
15.6 Drawing a Control 444
15.7 Backed-up Windows 451
15.8 Backed-up-Behind Windows 452
15.9 Dimmed and Invisible Controls 454
Summary 455

16 Dialogs 457
16.1 What Is a Dialog? 457
16.2 Simple Dialogs 459
16.3 Complex Dialogs 462
16.4 Single-Page Dialogs 464
16.5 Multi-Page Dialogs 466
16.6 Dialog APIs 472
16.7 Stock Controls for Dialogs 475
16.8 Custom Controls in Dialogs 477
Summary 480

17 Graphics for Display 481
17.1 Drawing Basics 482
17.2 The `CGraphicsContext` API 486

FIFTH THIRD EX1062
PART 2 of 3
Page 11 of 379

17.3 Drawing and Redrawing 492
17.4 Drawing Controls 499
17.5 Sharing the Screen 502
17.6 Support for Drawing in `CCoeControl` 514
17.7 Special Effects 521
17.8 Window Server Features 526
17.9 Device- and Size-Independent Graphics 529
Summary 556

18 Graphics for Interaction 557
18.1 Key, Pointer and Command Basics 558
18.2 User Requirements for Interaction 559
18.3 Some Basic Abstractions 561
18.4 Processing Key Events 567
18.5 Processing Pointer Events 571
18.6 Window Server and Control Environment APIs 576
Summary 582

19 Plug-ins and Extensibility 583
19.1 System Services 583
19.2 What Is a Plug-in? 587
19.3 The ECOM Library 593
19.4 Plug-ins in Symbian OS 603
Summary 604

20 Communications and Messaging Services 605
20.1 Communications in Noughts and Crosses 605
20.2 Communication Between Controller and Transport 606
20.3 Serial Communications 609
20.4 Socket-based Communications 615
20.5 Messaging 632
20.6 Security 646
Summary 649

21 Multimedia 651
21.1 The Multimedia Framework 651
21.2 The Image Conversion Library 682
21.3 Camera API 699
21.4 Tuner API 706
Summary 712

22 Introduction to SQL RDBMS 715
22.1 Overview of RDBMS 715
22.2 SQL Basics 716

22.3 Symbian SQL Server Component Architecture 729
22.4 Symbian SQL Error Codes 751
Summary 753

Appendix: Developer Resources 755

References 761

Index 763

FIFTH THIRD EX1062
PART 2 of 3
Page 13 of 379

About this book

Symbian OS C++ for Mobile Phones Volume 3 draws on the experience of Symbian's own engineers to provide a thorough grounding in writing C++ applications for mobile phones that use Symbian OS version 9.0 and beyond. It won't teach you *everything* you need to know about developing Symbian OS applications – no single book could do that. However, it will take you a long way along the road to being an effective Symbian OS developer, and give you a deep understanding of the fundamental principles upon which Symbian OS is based. The text is complemented with a specially developed suite of examples.

The book is broadly organized into four sections:

- **Section one** (Chapters 1 to 6) introduces Symbian OS itself, and describes the basic building blocks and usage patterns.
- **Section two** (Chapters 7 to 11) explains the core concepts, resources, APIs and programming idioms that you need to create, test and publish a simple GUI (graphical user interface) application.
- **Section three** (Chapters 12 to 18) provides a detailed description of the use of the Symbian OS graphical user interface to create non-trivial stand-alone applications. It looks deeply into the effective use of the available range of graphics APIs and helps you to ensure that your application code is as device-independent as possible.
- **Section four** (Chapters 19 to 22) provides an introduction to some of the most significant and useful of the Symbian OS system services. Starting with a discussion of extensibility and the use of plug-ins, this section continues with practically-based descriptions of the communications, multimedia and database services.

FIFTH THIRD EX1062
PART 2 of 3
Page 14 of 379

Symbian OS is used in a variety of phones with widely differing screen sizes. Some have full alphanumeric keyboards, some have touch-sensitive screens and some have neither. As far as possible, the material in this book is independent of any particular user interface. However, real applications run on real phones so, where necessary, we have chosen to use concrete examples based on both the S60 and UIQ user interfaces. Wherever relevant, the text explains the principal differences between these two user interfaces. This kind of information is invaluable for anyone who wishes to create versions of an application to run on a variety of Symbian OS phones.

Symbian OS C++ for Mobile Phones Volume 3 complements Symbian OS software development kits. When you've put this book down, the UIQ and S60 SDKs will be your first resource for reference information on the central Symbian OS APIs that we cover here. For more specialized and up-to-date information relating to a specific mobile phone, you will probably need to refer to a phone-specific SDK, available from the relevant manufacturer.

The SDKs contain valuable guide material, examples and source code, which together add up to an essential developer resource. We've pointed to these where they tie in with the book content. But as a general rule, look in the SDK anyway: you'll usually find additional information that explains things further than we could in this one book.

Who Is This Book For?

If you've programmed, at any level, in C++, it's for you. As a real and comprehensive system written in C++ from the ground up, and targeted at the high-growth area where computers and mobile communications converge, Symbian OS gives you unparalleled opportunities in mass-market, enterprise and system programming.

Besides C++ programmers, this book is of interest to other audiences:

- any other programmer or manager looking to exploit the potential of mobile solutions with Symbian OS technology
- consultants, trainers and authors thinking of basing their activity on Symbian OS technology
- anyone with an interest in system design, since Symbian OS is a full and interesting example in its own right.

Conventions

To help you get the most from the text and keep track of what's happening, we've used a number of conventions throughout the book.

FIFTH THIRD EX1062
PART 2 of 3
Page 15 of 379

These boxes hold important, not-to-be forgotten information that is directly relevant to the surrounding text.

We use several different fonts in the text of this book:

When we refer to words you use in your code, such as variables, classes and functions, or refer to the name of a file, we use this style: `iEikonEnv`, `ConstructL()`, or `e32base.h`.

URLs are written like this: ***www.symbian.com/developer***

And when we list code, or the contents of files, we use the following convention:

```
Lines that show concepts directly related to the surrounding text are
shown on a gray background
```

We show commands typed at the command line like this:

```
abld build winscw udeb
```

FIFTH THIRD EX1062
PART 2 of 3
Page 16 of 379

Glossary

ACID	A database operation should be atomic, consistent, isolated, durable
ACS Publisher ID	Class 3 ID certificate
Actor	Something or someone (a system or end-user) with which or whom a use case scenario interacts
ANSI	American National Standards Institute
AppUi	A class that forms the core of an application
Automatic variable	a variable that is created automatically when it is required and destroyed when it goes out of scope
BLOB	Binary Large Object
Capabilities	API protection within Symbian OS
Chunk	A mechanism by which the kernel (memory model) allocates and manages memory, contains physical RAM pages mapped to virtual addresses
CLOB	Character Large Object
CONE	A control environment that provides the basic framework for controls
Context switch	A task-switching facility to switch between programs and keep track of the execution environment for the first program
Control	provides the principal means of interaction between an application and the user; each application view is a control, which forms the basis of all dialogs and menu panes

Co-operative multitasking	A multitasking environment in which multiple programs running on a CPU voluntarily yield control of the processor to each other
DBMS	Database Management System
DFC	A deferred function call that forms part of the driver non-ISR processing
Dialog	A specialized window-owning compound control.
DLL	Dynamic Linked Library
ECOM	A system service developed with the express purpose of enabling application extensibility
EKA2	The Symbian OS kernel architecture
Eshell	A simple command-line-mode shell
Event-driven Application	An application that is always waiting for the user to interact with the device and then carries out some operation in response to that interaction
FEP	Front-end processor
FIFO	First-in, first-out order
GC	Graphics context
GUI	Graphical user interface
IMEI	A unique number used by the GSM network to identify valid devices
IPC	Inter-Process Communication
ISR	Interrupt-Service Routine
MBM	Multi-bitmap file format; a graphics format used by Symbian OS
Metrowerks Target	A Symbian OS application that runs inside the OS and resident kernel and provides debugging services over serial communications (MetroTRK)
Non-pre-emptive Multitasking	see co-operative multitasking
Panic	A run-time failure in a program
Plug-in	A computer program that interacts with a main (or host) application (a web browser or an email program, for example) to provide a certain, usually very specific, function on-demand.
Pre-emptive Multitasking	A multitasking environment in which the operating system determines when and for how long a program has control of the processor

FIFTH THIRD EX1062
PART 2 of 3
Page 18 of 379

Process	A running instance of a program; unit of memory protection
RDBMS	A relational-database management system
Real-time Nanokernel	The core of the modern Symbian OS kernel architecture (EKA2)
Round robin	an operating-system scheduling algorithm which allocates equal processing time to all active processes
Scalable vector graphics	A format for icons, introduced by S60 3rd edition
SID	Secure ID
Store	A collection of streams, generally used to implement the persistence of objects
Stream	An external representation of one or more objects
Thread	The basic unit of execution – a sequence of instructions that can be executed
Two-phase construction	A mechanism that separates the safe constructions (which can be put into the constructor) from the unsafe constructions
Use case	A complete sequence of actions, initiated by an actor, that represents a particular way of using a system; often used for test scenarios
View server	Aerver with which each application may register its views
View	A class that implements an interface to the view architecture
Window server	A server that ensures that the correct window or windows are displayed, managing overlaps and exposing and hiding windows as necessary
XML	Extensible Markup Language

FIFTH THIRD EX1062
PART 2 of 3
Page 19 of 379

1

Getting Started

It seems to be traditional to start a book on computer programming with a 'Hello World' example and, although this book is more about an operating system than a programming language, we are following that tradition. In the process we introduce you to the emulator and to the tools for building C++ programs, so that by the end of the chapter you will know how to build and run a Symbian OS application. We don't get too involved in describing Symbian OS programming conventions, API functions, and so forth; instead, we concentrate on the tools you need and how to use them, leaving the details until later chapters.

First we briefly describe the emulator. Most Symbian OS software is developed first on the emulator and only then on real target hardware. The emulator also includes a number of Symbian OS applications, and so mimics a real Symbian OS phone very closely. You will need to become familiar with the emulator and in the process we can take a look at the various graphical user interfaces (GUIs) used by Symbian OS.

Then we create a program. The easiest things to build are text-mode console programs, so that's the form of the classic 'Hello World' application that we use. We demonstrate how to compile it for the emulator, and how to launch it using the Carbide.c++ IDE.

1.1 Using the Emulator

The emulator is a fundamental tool for all the Symbian OS SDKs, so it's vital that you get to know how to use it.

If you are a newcomer to Symbian OS, the emulator offers an opportunity to get to know some Symbian OS basics from a user's perspective, so we look at these straight away. Later, you'll want to learn to make

FIFTH THIRD EX1062
PART 2 of 3
Page 20 of 379

2

A System Introduction to Symbian OS

Before we can start writing programs for Symbian OS, we need to appreciate the paradigms, idioms and philosophy that make up the architecture of Symbian OS and its C++ frameworks. So far, we've seen the basics of how to build programs for Symbian OS. Now we need some background information in order to understand how to write them.

In this chapter we'll introduce as many concepts as we need for the following chapters, without getting deeply involved in C++ code. All these issues are fundamental for appreciating the system design of Symbian OS and will be essential background as we move to look at areas such as coding conventions, the user library and other basic APIs in the next few chapters.

Since this chapter discusses the fundamentals of the system, the discussion is divided between the operating system basics and the framework principles that encompass them. Even though it is very rare for a domain-specific programming framework to be architected and developed synchronously with its underlying operating system, the two influence each other throughout and it is difficult to appreciate, or even explain, one without the other.

2.1 Symbian OS Basics

Symbian OS is intended to run on open (to third-party installable applications), small, battery-powered portable computers, that is, modern advanced mobile phones. This profoundly affects the design of the Symbian OS software system.

Symbian OS is designed for 32-bit CPUs, running at low speeds relative to those in desktops or workstations. Today's Symbian OS systems are based on XScale, ARM9 and ARM11-based CPUs, while older devices are powered by the ARM7 family of CPUs.

FIFTH THIRD EX1062
PART 2 of 3
Page 21 of 379

9

Platform Security and Publishing Applications

This chapter deals with preparing your application for distribution. We discuss information related to building an application for, and running it on, a target machine, including giving your application an icon and building a distributable SIS file from a PKG file. There is also an introduction to platform security, so that you can successfully prepare your application for the Symbian Signed program.

9.1 Releasing an Application

In order to release your application, some important steps must be taken. These steps do not require you to add more C++ code, but are considered critical to ensuring the success of an application in the mass market. The extra work involved focuses on things such as preparing an icon for the application, building it as an installable application and authorizing it under the Symbian Signed program. The last step is especially important with the introduction of platform security in Symbian OS v9.

Symbian OS v9 is a major evolution. It is specifically geared to target mid-range phones in tens and hundreds of millions of units – offering even larger opportunities for developers to market their products. As such, it includes a significant number of new features and improvements to enable key emerging technologies required by advanced, open mobile phones for years to come. In order to provide comprehensive support for new functionality such as Digital Rights Management, Device Management and Enterprise-grade data handling, large changes have been made to the

12

A Simple Graphical Application

After a series of chapters describing individual aspects of Symbian OS, it is time to pull a few threads together. In Chapter 11, we described a minimal GUI application that simply showed how the UI framework and the application UI fit together. Now we look at a complete, simple, but non-trivial example application that forms a basis for the examples used in several of the following chapters.

In this chapter, we cover some aspects of application design and introduce the idea of how to program a user interface that does not assume a particular screen size. This latter point is important if you want to target your application at multiple user interfaces, as it helps speed up your porting efforts. In addition, we show how an application can use the principles described in Chapter 7 to save and restore its persistent data.

Before looking at the application's structure and behavior in more detail, we start with a general introduction to the Noughts and Crosses (otherwise known as Tic-Tac-Toe) application. Figure 12.1 shows the application running on a phone using the S60 user interface.

This application runs a game of noughts and crosses in which two players take turns to make their moves. The object of the game is to place three noughts (or crosses) in any row, column or diagonal before your opponent manages to do so.

We've implemented the game in a way that minimizes the differences needed to run on phones using the S60 and UIQ user interfaces. Depending on the nature of the phone it is running on, you can make a move either by using a combination of the cursor and pushbutton keys or by tapping on the screen.

FIFTH THIRD EX1062
PART 2 of 3
Page 23 of 379

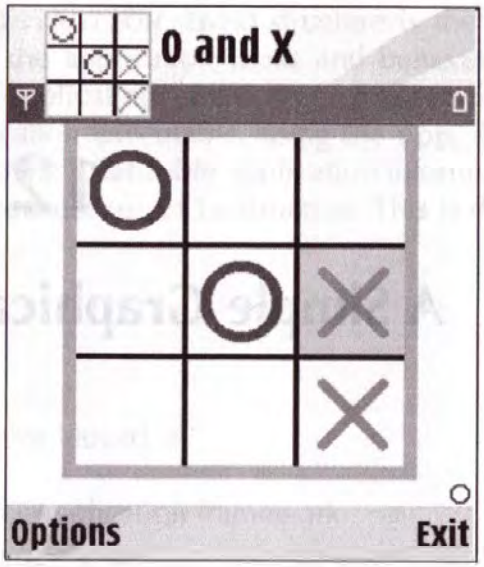


Figure 12.1 The Noughts and Crosses application

We've also kept the logic relating to the game itself very simple so that the bulk of the code illustrates issues of a general nature, relevant to most applications. In particular, we've designed the layout of the display in a way that can easily be adapted to show a rectangular grid with any number of tiles and to display on phones with a variety of screen formats. The grid layout is defined by a set of constants in `oandxdefs.h`:

```
static const TInt KTilesPerRow   = 3;
static const TInt KTilesPerCol   = 3;
static const TInt KNumberOfTiles = (KTilesPerRow*KTilesPerCol);
```

All the application's layout and drawing code is based on these values. By changing them, the layout can be adapted for games using differently sized and shaped boards.

The same header file also defines values that are specific to the Noughts and Crosses game, including the constant `KTilesPerSide` (which, unlike the earlier values, assumes that the board layout is square) and the states that can be associated with each tile:

```
static const TInt KTilesPerSide = 3;

enum TTileState
  {
  ETileBlank  = 0,
  ETileNought = 1,
  ETileCross  = 4
  };
```

FIFTH THIRD EX1062
PART 2 of 3
Page 24 of 379

The state values are chosen so that `ETileCross` is greater than the product of `ETileNought` with either `KTilesPerRow` or `KTilesPerCol`, which helps with the calculation of a winner.

This file also defines the height of the application's status window, used to show whose turn it is to play:

```
static const TInt KStatusWinHeight = 20; // in pixels
```

In Chapter 11 you saw that the simplest Symbian OS graphical application needs instances of four classes: the application, the document, the application UI and a view. For this game, we've added two more classes: a controller and a model, or engine. Since the game control logic is so simple, we could have implemented it entirely within the application UI class, rather than defining separate controller and engine classes. However, we've chosen to add these classes from the start, in preparation for the more complex form of the game that is developed in Chapter 20.

The overall structure of the game is shown in Figure 12.2.

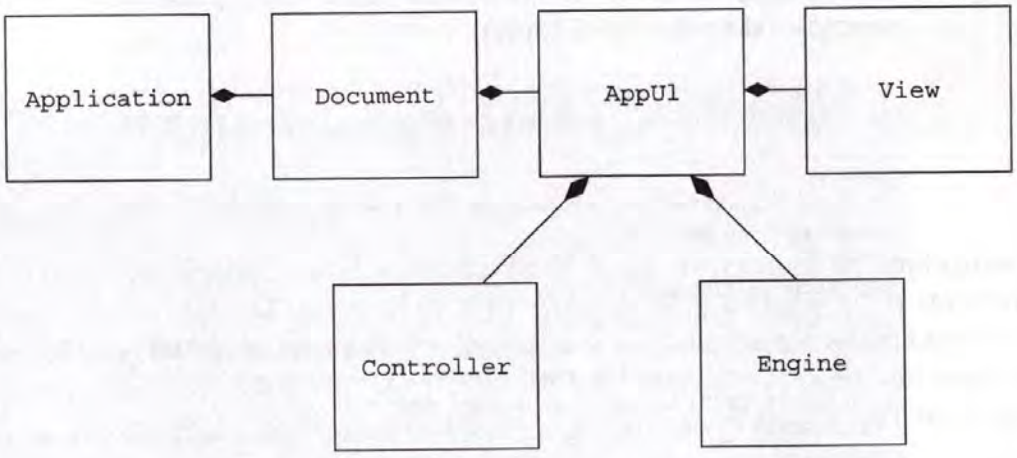


Figure 12.2 Structure of the Noughts and Crosses application

We cover the S60 version of the game first and then look at the changes that are needed to create the UIQ version.

12.1 Implementing the Game on S60

The Application Class

The S60 class definition for the Noughts and Crosses application class is listed below. As you can see, it is very similar to the corresponding class in Chapter 11's basic application: it implements `AppDllUid()`, to report the application's UID to the application framework, and `CreateDocumentL()`.

FIFTH THIRD EX1062
PART 2 of 3
Page 25 of 379

```
class COandXApplication : public CAknApplication
  {
public:
  // From CApaApplication
  TUid AppDllUid() const;

protected:
  // From CEikApplication
  CApaDocument* CreateDocumentL();
  };
```

This is fairly typical for applications. It is rare for an application class to override any of the other application class member functions.

The Document Class

The class definition for the Noughts and Crosses document class, listed below, provides the implementation of `CreateAppUiL()`, which is always required. In addition, it implements persistent storage of the game's state in the `StoreL()` and `RestoreL()` functions, using the principles described in Chapter 7.

```
class COandXDocument : public CAknDocument
  {
public:
  static COandXDocument* NewL(CEikApplication& aApp);
  virtual ~COandXDocument();

  // From CEikDocument
  CEikAppUi* CreateAppUiL();
  void StoreL(CStreamStore& aStore, CStreamDictionary& aStreamDic) const;
  void RestoreL(const CStreamStore& aStore,
      const CStreamDictionary& aStreamDic);
  CFileStore* OpenFileL(TBool aDoOpen, const TDesC& aFilename, RFs& aFs);
private:
  COandXDocument(CEikApplication& aApp);
private:
  COandXAppUi* iAppUi;
  };
```

The implementations of `StoreL()` and `RestoreL()` delegate to the application UI class the actual task of storing and restoring the application's data, but they retain the responsibility for associating the application's UID with the ID of the stream in which the data is stored.

```
void COandXDocument::StoreL(CStreamStore& aStore,
            CStreamDictionary& aStreamDic) const
  {
  TStreamId id = iAppUi->StoreL(aStore);
```

FIFTH THIRD EX1062
PART 2 of 3
Page 26 of 379

```
  aStreamDic.AssignL(KUidOandXApp, id);
  }

void COandXDocument::RestoreL(const CStreamStore& aStore,
                    const CStreamDictionary& aStreamDic)
  {
  TStreamId id = aStreamDic.At(KUidOandXApp);
  iAppUi->RestoreL(aStore, id);
  }
```

The application framework provides support for an application to save its document data in an application-specific direct file store and, in UIQ, it is sufficient just to implement the two functions listed above.

In contrast, S60 applications are assumed not to be document-based and the default behavior is that the application framework never calls these two functions. In order to restore the support, you need to override the `OpenFileL()` function to call `CEikDocument`'s implementation of `OpenFileL()`:

```
CFileStore* COandXDocument::OpenFileL(TBool aDoOpen,
                  const TDesC& aFilename, RFs& aFs)
  {
  return CEikDocument::OpenFileL(aDoOpen, aFilename, aFs);
  }
```

The Application UI Class

The application UI class normally forms the core of an application, and the `COandXAppUi` class is no exception. In addition to its standard role of handling much of the interaction with the user, `COandXAppUi` owns the application's view, controller and engine instances and supplies the central logic for saving and restoring the application's persistent data.

```
class COandXAppUi : public CAknAppUi
  {
public:
  COandXAppUi();
  virtual ~COandXAppUi();

  // New functions
  void ReportWhoseTurn();
  void ReportWinnerL(TInt aWinner);

  // From CEikAppUi, for persistent data
  TStreamId StoreL(CStreamStore& aStore) const;
  void RestoreL(const CStreamStore& aStore, TStreamId aStreamId);
  void ExternalizeL(RWriteStream& aStream) const;
  void InternalizeL(RReadStream& aStream);

private:
  // From MEikMenuObserver
```

```
  void DynInitMenuPaneL(TInt aResourceId, CEikMenuPane* aMenuPane);
  // From CEikAppUi
  void HandleCommandL(TInt aCommand);
  void ConstructL();

public:
  // AppUi owns the engine, controller and application view.
  COandXEngine* iEngine;
  COandXController* iController;
private:
  COandXAppView* iAppView;
  // Set if the application view was added to the control stack.
  TBool iStacked;
  };
```

As you can see from the above class definition, part of the user interaction consists of reporting significant events such as whose turn it is and, at the end of a game, who (if anyone) has won. In the case of the S60 version, the application UI class also handles all aspects of using menu commands, including dynamically setting the menu content according to the current state of the game. We have more to say about using menus later in this chapter.

The second-phase constructor and the destructor make it clear that the application UI class owns the engine, the controller and the view, which are all created in a leave-safe way:

```
void COandXAppUi::ConstructL()
  {
  BaseConstructL(EAknEnableSkin);
  iEngine = COandXEngine::NewL();
  iController = COandXController::NewL();
  iAppView = COandXAppView::NewL(ClientRect());
  AddToStackL(iAppView); // Enable keypresses to the view
  iStacked = ETrue;
  ReportWhoseTurn();
  }

COandXAppUi::~COandXAppUi()
  {
  if (iStacked)
    {
    RemoveFromStack(iAppView);
    }
  delete iAppView;
  delete iController;
  delete iEngine;
  }
```

It is worth noting that, it is the responsibility of the application UI class to ensure that the view is added to the control stack, via a call to `AddToStackL()`, to ensure that it has the opportunity to receive and process keypresses. Since this call can fail, it is necessary to set a flag if

FIFTH THIRD EX1062
PART 2 of 3
Page 28 of 379

it succeeds, so that the flag can be tested (in this case, since there is only one view, in the destructor) before the view is removed from the control stack.

This application uses two different ways to report events to the user. Reporting whose turn it is to play, which happens between every move in the game, is delegated to the view:

```
void COandXAppUi::ReportWhoseTurn()
  {
  iAppView->ShowTurn();
  }
```

If either player wins, which obviously happens no more than once per game, the application UI class reports the winner by means of a UI-specific dialog that displays the appropriate text, read from the application's resource file. The S60 version uses an information note:

```
void COandXAppUi::ReportWinnerL(TInt aWinner)
  {
  TBuf<MaxInfoNoteTextLen> text;
  iEikonEnv->ReadResource(text, aWinner==ETileCross
                ? R_OANDX_X_WINS : R_OANDX_O_WINS);
  CAknInformationNote* infoNote = new (ELeave) CAknInformationNote;
  infoNote->ExecuteLD(text);
  }
```

The application's persistent data is saved and restored via the `StoreL()` and `RestoreL()` functions of the application UI class which, you may recall, are called from the application's document class.

```
TStreamId COandXAppUi::StoreL(CStreamStore& aStore) const
  {
  RStoreWriteStream stream;
  TStreamId id = stream.CreateLC(aStore);
  stream << *this; // alternatively, use ExternalizeL(stream)
  stream.CommitL();
  CleanupStack::PopAndDestroy();
  return id;
  }

void COandXAppUi::RestoreL(const CStreamStore& aStore,
                                  TStreamId aStreamId)
  {
  RStoreReadStream stream;
  stream.OpenLC(aStore, aStreamId);
  stream >> *this; // alternatively use InternalizeL(stream)
  CleanupStack::PopAndDestroy();
  }
```

`Store()` and `Restore()`, respectively, call the `ExternalizeL()` and `InternalizeL()` functions of the application UI class via the

FIFTH THIRD EX1062
PART 2 of 3
Page 29 of 379

<< and >> operators. As you can see from the following listing, the application UI class itself has no persistent data. The functions deal directly with the view's data, which is simply the ID of the control that currently has focus, and leave the engine and the controller to take care of their own data.

```
void COandXAppUi::ExternalizeL(RWriteStream& aStream) const
  {
  iEngine->ExternalizeL(aStream);
  iController->ExternalizeL(aStream);
  aStream.WriteInt8L(iAppView->IdOfFocusControl());
  }

void COandXAppUi::InternalizeL(RReadStream& aStream)
  {
  iEngine->InternalizeL(aStream);
  iController->InternalizeL(aStream);
  ReportWhoseTurn();
  iAppView->MoveFocusTo(aStream.ReadInt8L());
  }
```

The S60 application UI class is also responsible for handling menu commands, via the `DynInitMenuPaneL()` and `HandleCommandL()` functions. We discuss them – and the differences between the ways they are handled in S60 and UIQ – later in this chapter.

Finally, it is worth pointing out that we provide access to the controller and engine via global static functions, declared (in `oandxappui.h`) as:

```
GLREF_C COandXAppUi* OandXAppUi();
GLREF_C COandXController& Controller();
GLREF_C COandXEngine& Engine();
```

The implementations of these functions are:

```
GLDEF_C COandXAppUi* OandXAppUi()
  {
  return static_cast<COandXAppUi*>(CEikonEnv::Static()->AppUi());
  }

GLDEF_C COandXController& Controller()
  {
  return *OandXAppUi()->iController;
  }

GLDEF_C COandXEngine& Engine()
  {
  return *OandXAppUi()->iEngine;
  }
```

We have chosen to use global functions in this example because they accurately represent the intention that these functions should be

FIFTH THIRD EX1062
PART 2 of 3
Page 30 of 379

accessible from anywhere in the program. Furthermore, they do not need verbose source text to call them, which is useful in example code. We would not use them in production code, since there is no control of when and from where they are called. Additionally, calling a global function involves accessing thread-local storage, which is slow compared with other function calls. We could improve the efficiency by implementing them as static members of a class, but this relies on the availability of writable static data and so would not work on all phones. The best solution would be to implement them as normal member functions of, for example, the application UI class and pass a (non-owning) reference to the constructor of each class that needs to access them. We have not done that in this example because it would add a significant amount of incidental complexity to each of the constructors concerned.

The Controller Class

The controller class, whose definition is listed below, is responsible for managing the state of the game and its logic, including updating the data in the engine class.

```
class COandXController : public CBase
  {
public:
  static COandXController* NewL();
  virtual ~COandXController();
enum TState
  {
  ENewGame, EPlaying, EFinished
  };
  // state
  inline TBool IsNewGame() const;
  // game control
  void Reset();
  // stream persistence
  void ExternalizeL(RWriteStream& aStream) const;
  void InternalizeL(RReadStream& aStream);

  TBool HitSquareL(TInt aIndex);
  TBool IsCrossTurn() const;
  void SwitchTurn();
private:
  void ConstructL();
private:  // private persistent state
  TState iState;
  TBool iCrossTurn;
  };
```

In this stand-alone, non-communicating, version, the game state is simply one of the three values `ENewGame`, `EPlaying` or `EFinished`, together with a flag that indicates which player is next to play. The only externally accessible state-query functions needed are `IsCrossTurn()`,

FIFTH THIRD EX1062
PART 2 of 3
Page 31 of 379

which simply returns the value of `iCrossTurn`, and `IsNewGame()`, coded as:

```
inline TBool COandXController::IsNewGame() const
  { return iState==ENewGame; }
```

The central logic of the controller is contained in the `HitSquareL()` function:

```
TBool COandXController::HitSquareL(TInt aIndex)
  {
  if (iState == EFinished)
    {
    return EFalse;
    }
  if (iState == ENewGame)
    {
    iState = EPlaying;
    }
  if (Engine().TryMakeMove(aIndex, IsCrossTurn()))
    {
    SwitchTurn();
    TInt winner = Engine().GameWonBy();
    if (winner)
      {
      iState = EFinished;
      OandXAppUi()->ReportWinnerL(winner);
      }
    return ETrue;
    }
  return EFalse;
  }
```

This function is called from the application's board view whenever a player attempts to make a move. It disallows the move if the game state is `EFinished`; if this is the first move in the game (i.e. the game state is `ENewGame`), it sets the state to `EPlaying`. If the game is in play, it calls the `TryMakeMove()` function of the engine class to check if the move is valid; if so, it records the move and switches whose turn it is. A call to the `GameWonBy()` function of the engine class checks if the last move resulted in a win by either player and, if it did, the game state is set to `EFinished` and a call to the `ReportWinner()` function of the application UI class displays the result.

In its current form, `HitSquareL()` does not check for, or report, a drawn game. If there is no winner, the game continues until all tiles contain a nought or a cross (or a player uses a menu item to start a new game). The functionality for a drawn game is added in the communicating version of the game, described in Chapter 20.

The controller is reset for a new game by calling `Reset()`, which cancels the current game, clears the board (by calling the `Reset()` function of the engine class) and sets Noughts as the current player:

FIFTH THIRD EX1062
PART 2 of 3
Page 32 of 379

```
void COandXController::Reset()
  {
  Engine().Reset();
  iState = ENewGame;
  if (IsCrossTurn())
    {
    SwitchTurn();
    }
  }
```

The `SwitchTurn()` function simply negates the flag held in `iCrossTurn` and calls the `ReportWhoseTurn()` function of the application UI class:

```
void COandXController::SwitchTurn()
  {
  iCrossTurn = !iCrossTurn;
  OandXAppUi()->ReportWhoseTurn();
  }
```

Persistence of the state of the controller class is handled by the `ExternalizeL()` and `InternalizeL()` functions, which are called from the application UI class. Their actions are, respectively, to write and read the controller's member data to or from the passed stream:

```
void COandXController::ExternalizeL(RWriteStream& aStream) const
  {
  aStream.WriteUint8L(iState);
  aStream.WriteInt8L(iCrossTurn);
  }

void COandXController::InternalizeL(RReadStream& aStream)
  {
  iState = static_cast<TState>(aStream.ReadUint8L());
  iCrossTurn = static_cast<TBool>(aStream.ReadInt8L());
  }
```

The controller is created via a call to its static `NewL()` function, which is coded as:

```
COandXController* COandXController::NewL()
  {
  COandXController* self=new(ELeave) COandXController;
  CleanupStack::PushL(self);
  self->ConstructL();
  CleanupStack::Pop();
  return self;
  }
```

FIFTH THIRD EX1062
PART 2 of 3
Page 33 of 379

The second-phase constructor is simply:

```
void COandXController::ConstructL()
  {
  Reset();
  }
```

We have coded it in this way to preserve the standard form of class construction, as described in Chapter 4. However, since – in this case – `ConstructL()` does not leave, it is worth pointing out that we could have simplified matters by not defining a `ConstructL()` function and doing everything in the `NewL()` function.

The Engine Class

The engine class, whose class definition is listed below, represents the game data and contains the functions that operate on that data. Since Noughts and Crosses is a simple game, the engine class is correspondingly straightforward. The class itself has no knowledge of the overall state of the game, nor of whose turn it is to make a move. As you saw in the previous section, both of these are the responsibility of the controller.

```
class COandXEngine : public CBase
  {
public:
  static COandXEngine* NewL();
  virtual ~COandXEngine();

  void Reset();
  TInt TileStatus(TInt aIndex) const;
  TBool TryMakeMove(TInt aIndex, TBool aCrossTurn);
  TTileState GameWonBy() const;
  // persistence
  void ExternalizeL(RWriteStream& aStream) const;
  void InternalizeL(RReadStream& aStream);
private:
  COandXEngine();
  TInt TileState(TInt aX, TInt aY) const;
private:
  TFixedArray<TTileState, KNumberOfTiles> iTileStates;
  };
```

The current state of the board is represented by a simple array of tile states, one for each component tile. Each tile state contains one of the three values `ETileBlank`, `ETileNought` or `ETileCross`.

The key functions are `TryMakeMove()` and `GameWonBy()`, both of which are called from the controller. `TryMakeMove()` only allows a move to be made if the corresponding tile is initially blank. If the move is allowed, the function sets the tile-state variable corresponding to the

FIFTH THIRD EX1062
PART 2 of 3
Page 34 of 379

tile to either `ETileCross` or `ETileNought`, depending on the value passed in `aCrossTurn`.

```
TBool COandXEngine::TryMakeMove(TInt aIndex, TBool aCrossTurn)
  {
  if (iTileStates[aIndex] == ETileBlank)
    {
    iTileStates[aIndex] = aCrossTurn ? ETileCross : ETileNought;
    return ETrue;
    }
  return EFalse;
  }
```

`GameWonBy()` checks if there is a winner and, if so, whether the winner is the Nought player or the Cross player. It does so by determining if there is a horizontal, vertical or diagonal line of three noughts or three crosses. If there is such a line, it returns either `ETileNought` or `ETileCross`; otherwise it returns zero, to indicate that, as yet, there is no winner.

```
TTileState COandXEngine::GameWonBy() const
  {
  const TInt KNoughtWinSum = KTilesPerSide * ETileNought;
  const TInt KCrossWinSum = KTilesPerSide * ETileCross;

  // is there a row or column of matching tiles?
  for (TInt i = 0; i < KTilesPerSide; ++i)
    {
    TInt rowSum = 0;
    TInt colSum = 0;
    for (TInt j = 0; j < KTilesPerSide; ++j)
      {
      rowSum += TileState(j, i);
      colSum += TileState(i, j);
      }
    if (rowSum == KNoughtWinSum || colSum == KNoughtWinSum)
      return ETileNought;
    if (rowSum == KCrossWinSum || colSum == KCrossWinSum)
      return ETileCross;
    }
  // is there a diagonal of matching tiles?
  TInt blTrSum = 0;        // bottom left to top right
  TInt tlBrSum = 0;        // top left to bottom right
  for (TInt i = 0; i < KTilesPerSide; ++i)
    {
    tlBrSum += TileState(i, i);
    blTrSum += TileState(i, KTilesPerSide - 1 - i);
    }
  if (blTrSum == KNoughtWinSum || tlBrSum == KNoughtWinSum)
    return ETileNought;
  if (blTrSum == KCrossWinSum || tlBrSum == KCrossWinSum)
    return ETileCross;
  return ETileBlank;  // No winner
  }
```

As with the controller class, the `ExternalizeL()` and `InternalizeL()` functions of the engine class, called from the application UI class, simply write and read the state – in this case, the values of each of the board's tiles – to and from the appropriate stream:

```
void COandXEngine::ExternalizeL(RWriteStream& aStream) const
  {
  for (TInt i = 0; i<KNumberOfTiles; i++)
    {
    aStream.WriteInt8L(iTileStates[i]);
    }
  }

void COandXEngine::InternalizeL(RReadStream& aStream)
  {
  for (TInt i = 0; i<KNumberOfTiles; i++)
    {
    iTileStates[i] = static_cast<TTileState>(aStream.ReadInt8L());
    }
  }
```

Like the controller class, the engine class is constructed by means of a call to its static `NewL()` function and, again, there are no potentially leaving calls to be made from a second-phase constructor. For the engine class, we've chosen to illustrate an alternative way of implementing construction in such a case, where the non-leaving calls are made from the constructor rather than from the `NewL()` function:

```
COandXEngine* COandXEngine::NewL()
  {
  return new(ELeave) COandXEngine;
  }

COandXEngine::COandXEngine()
  {
  Reset();
  }
```

The View Class

Views and the view architecture are discussed in Chapter 14 and the detailed behavior of controls is described in Chapters 15, 17 and 18. In this chapter, therefore, we only briefly describe the general aspects of controls and views, concentrating on those features that are specific to the Noughts and Crosses application.

An application's view is an instance of a window-owning control (as defined in Chapter 15) and derives, directly or indirectly, from the `CCoeControl` class. For the Noughts and Crosses application, the view is a compound control, with the view's area being tiled with a number of subsidiary controls.

FIFTH THIRD EX1062
PART 2 of 3
Page 36 of 379

We've chosen to implement the S60 view class by deriving directly from `CCoeControl`. S60 applications can derive from more specific classes, such as `CAknView`, but we decided not to for this application in order to reduce the differences between the S60 and UIQ versions, and because many of the S60-specific views are more suited to text displays. The view also inherits from an `MViewCmdHandler` interface class, which we describe later.

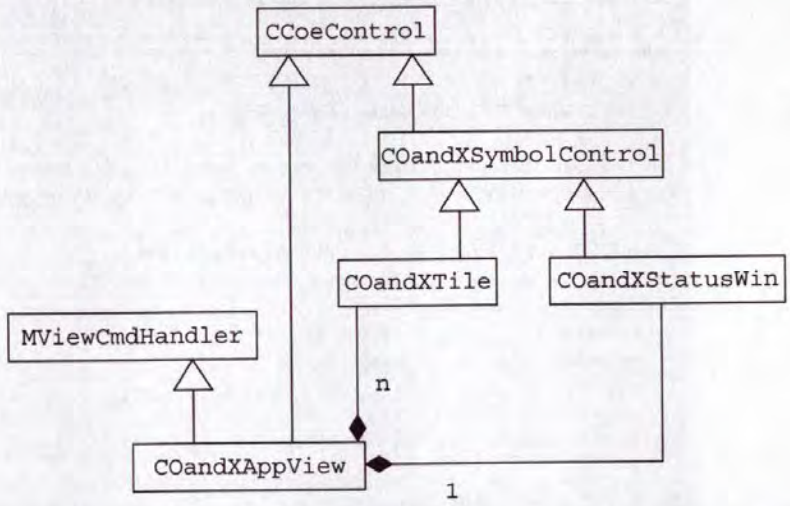


Figure 12.3 The View class of the Noughts and Crosses application

Figure 12.3 shows that all the components of the view are themselves also controls, in the sense that they all derive from `CCoeControl`. The view owns a single status-display control and a series of tiles.

Both the status display control and the tiles need to display either a nought or a cross, so we've chosen to implement them as subclasses of a generic symbol-drawing control, which knows how to draw those symbols. The two subclasses simply add the specification of the size and location, within their own area, of the symbol.

As you can see from the class definition of the symbol-drawing base class, its only member function is to draw a nought or a cross symbol:

```
class COandXSymbolControl : public CCoeControl
  {
protected:
  void DrawSymbol(CWindowGc& aGc, const TRect& aRect,
                                  TBool aDrawCross) const;
  };
```

The implementation of this function is:

```
void COandXSymbolControl::DrawSymbol(CWindowGc& aGc,
        const TRect& aRect, TBool aDrawCross) const
```

FIFTH THIRD EX1062
PART 2 of 3
Page 37 of 379

```
  {
  TRect drawRect(aRect);

  // Shrink by about 15%
  drawRect.Shrink(aRect.Width()/6, aRect.Height()/6);

  // Pen size set to just over 10% of the shape's size
  TSize penSize(aRect.Width()/9, aRect.Height()/9);
  aGc.SetPenSize(penSize);
  aGc.SetPenStyle(CGraphicsContext::ESolidPen);

  if (aDrawCross)
    {
    aGc.SetPenColor(KRgbGreen);

    // Cosmetic reduction of cross size by half the line width
    drawRect.Shrink(penSize.iWidth/2, penSize.iHeight/2);

    aGc.DrawLine(drawRect.iTl, drawRect.iBr);
    TInt temp;
    temp = drawRect.iTl.iX;
    drawRect.iTl.iX = drawRect.iBr.iX;
    drawRect.iBr.iX = temp;
    aGc.DrawLine(drawRect.iTl, drawRect.iBr);
    }
  else // draw a circle
    {
    aGc.SetPenColor(KRgbRed);
    aGc.SetBrushStyle(CGraphicsContext::ESolidBrush);
    aGc.DrawEllipse(drawRect);
    }
  };
```

You see this code again in Chapter 15, where the more detailed aspects of drawing this control are discussed; here, it is sufficient to note that it is written to be as independent as possible of the size or position of the symbol to be drawn. The fact that the same code successfully draws both the large symbols in the tiles and the much smaller symbols in the status area shows that this approach is both feasible and sensible.

The symbol is centered in the rectangle (which is always a square) passed to the function in `aRect`, but shrunk to leave a small border, and the pen width is set to match the size of the symbol. Note that the cross symbol is shrunk by a further small amount since, if the circles and crosses were both drawn to exactly the same size, the cross would look larger to the eye. This kind of attention to detail can make a large difference to the appearance of any application.

The tile and status window control classes both derive from `COandX-SymbolControl`:

```
class COandXTile : public COandXSymbolControl
  {
public:
  COandXTile();
```

```
  ~COandXTile();
  void ConstructL(RWindow& aWindow);
  // New function
  void SetOwnerAndObserver(COandXAppView* aControl);
  // From CCoeControl
  TKeyResponse OfferKeyEventL(const TKeyEvent& aKeyEvent,
                              TEventCode aType);
  TCoeInputCapabilities InputCapabilities() const;
protected:
  void FocusChanged(TDrawNow aDrawNow);
  void HandlePointerEventL(const TPointerEvent& aPointerEvent);
private:
  void Draw(const TRect& aRect) const;
  // New function
  void TryHitL();
private:
  MViewCmdHandler* iCmdHandler;
  };

class COandXStatusWin : public COandXSymbolControl
  {
public:
  static COandXStatusWin* NewL(RWindow& aWindow);
  ~COandXStatusWin();
private:
  COandXStatusWin();
  void ConstructL(RWindow& aWindow);
  void Draw(const TRect& aRect) const;
  }
```

Note that, even though S60 devices do not normally use touch screens, we include code that handles input from keys – via `OfferKeyEventL()` – and any pointer device – via `HandlePointerEventL()`. It is quite safe to do this: if a particular device does not support one or the other input device, the relevant code simply is not called. Including both options makes it easier to port the code to different devices.

The status window's `Draw()` function simply calculates the square in which to draw the symbol and then calls `DrawSymbol()`, obtaining the flag indicating which symbol to draw by calling the `IsCrossTurn()` function of the controller class:

```
void COandXStatusWin::Draw(const TRect& /*aRect*/) const
  {
  CWindowGc& gc = SystemGc();
  TRect boxRect = Rect();
  gc.Clear(boxRect);
  TInt boxHeight = boxRect.iBr.iY - boxRect.iTl.iY;
  boxRect.iTl.iX = boxRect.iBr.iX - boxHeight;
  DrawSymbol(gc, boxRect, Controller().IsCrossTurn());
  }
```

The `Draw()` function for each tile works in a similar way:

```
void COandXTile::Draw(const TRect& /*aRect*/) const
  {
  TTileState tileType;
  tileType = iCmdHandler->TileStatus(this);

  CWindowGc& gc = SystemGc();
  TRect rect = Rect();

  if (IsFocused())
    {
    gc.SetBrushColor(KRgbYellow);
    }
  gc.Clear(rect);
  if (tileType!=ETileBlank)
    {
    DrawSymbol(gc, rect, tileType==ETileCross);
    }
  }
```

In this case, the symbol to be drawn is determined by a call to the `TileStatus()` function of the view class, by a mechanism that is described later. The tile which currently has focus is drawn with a yellow background; the others are drawn with the default brush color, which is white.

During the game, a player places a nought or a cross in a particular tile either by tapping on it or by using cursor keys to move the highlight to that tile and pressing the phone's Selection or Action key. A pointer event is handled by the tile's `HandlePointerEventL()` function:

```
void COandXTile::HandlePointerEventL (const TPointerEvent& aPointerEvent)
  {
  if (aPointerEvent.iType == TPointerEvent::EButton1Down)
    {
    TryHitL();
    }
  }
```

A key event is handled by `OfferKeyEventL()`:

```
TKeyResponse COandXTile::OfferKeyEventL(const TKeyEvent& aKeyEvent,
                                        TEventCode aType)
  {
  TKeyResponse keyResponse = EKeyWasNotConsumed;
  if (aType!=EEventKey)
    {
    return keyResponse;
    }
  switch (aKeyEvent.iCode)
    {
    case EKeyOK:
```

```
      TryHitL();
      keyResponse = EKeyWasConsumed;
      break;
    default:
      keyResponse = EKeyWasNotConsumed;
      break;
    }
  return keyResponse;
  }
```

In both cases, this results in a call to the tile's `TryHitL()` function, which is implemented as:

```
void COandXTile::TryHitL()
  {
  if (iCmdHandler->TryHitSquareL(this))
    {
    DrawDeferred();
    }
  }
```

`TryHitL()` calls the `TryHitSquareL()` function of the view class, using the same mechanism that is also used to call `TileStatus()`.

The `COandXAppView` class definition has 19 member functions, not counting those that it inherits from its base classes, but only the two mentioned above are called from a tile. As an encapsulation aid, we have chosen to declare these two functions, as pure virtual functions, in an `MViewCmdHandler` interface class:

```
class MViewCmdHandler
  {
public:
  virtual TBool TryHitSquareL(const COandXTile* aControl) = 0;
  virtual TTileState TileStatus(const COandXTile* aControl) const = 0;
  };
```

The view class inherits from this interface and provides the concrete implementations, as indicated by the bold portions of the `COandXAppView` class definition:

```
class COandXAppView : public CCoeControl, public MCoeControlObserver,
                                                public MViewCmdHandler
  {
public:
  static COandXAppView* NewL(const TRect& aRect);
  virtual ~COandXAppView();

  // From CCoeControl
  TKeyResponse OfferKeyEventL(const TKeyEvent& aKeyEvent,
                                        TEventCode aType);
```

```
  // new functions
  void MoveFocusTo(const TInt aIndex);
  TInt IdOfFocusControl();
  void ShowTurn();
  void ResetView();

private:
  COandXAppView();
  void ConstructL(const TRect& aRect);

  void SwitchFocus(TInt aFromIndex, CCoeControl* aToControl);
  void DrawComps(TRect& aRect) const;
  COandXTile * CreateTileL();

  // From CCoeControl
  void Draw(const TRect& aRect) const;
  void SizeChanged();
  TInt CountComponentControls() const;
  CCoeControl* ComponentControl(TInt aIndex) const;

  // From MCoeControlObserver
  void HandleControlEventL(CCoeControl* aControl, TCoeEvent aEventType);

  // From MViewCmdHandler
  TBool TryHitSquareL(const COandXTile* aControl);
  TTileState TileStatus(const COandXTile* aControl) const;

private:
  RPointerArray<COandXTile> iTiles; // View owns the tiles
  COandXStatusWin* iStatusWin;      // and its own status window.
  TRect iBoardRect;  // Board area
  TRect iBorderRect; // Bounding rectangle for border
  TInt iTileSide;    // Tile dimension, allowing for line widths
                     // and border
  };
```

Note that the functions are declared public in the interface class, but private within `COandXAppView`. By providing each tile with an `MViewCmdHandler*` pointer to the enclosing view, we expose only those two functions (they are not even available via a `COandXAppView*` pointer).

The interface class is a useful way of encapsulating the interaction between the control and the view or, indeed, the rest of the program. The control does not care about any of the view's internal details, provided that it correctly handles the request to play in a particular square and the querying of a square's content. Similarly, the view does not care whether the command originated with a key event, a pointer event, or some other kind of event – it just requires these functions to be called at the right time with the right parameters.

This is a useful technique, which is used frequently within a variety of Symbian OS APIs, and it is a good one to adopt for your programs, especially if they are more complicated than the Noughts and Crosses application. As a further illustration, think about the way commands

FIFTH THIRD EX1062
PART 2 of 3
Page 42 of 379

reach `HandleCommandL()`. The menu bar uses an `MEikMenuObserver` for this and does not otherwise care about the large API of the `CEikAppUi` class. The button bar uses the `MEikCommandObserver` interface. Similarly, command buttons use an `MCoeControlObserver` interface, which the button bar implements by converting button events into application commands.

`TryHitSquareL()` simply calls the controller's `HitSquareL()` function, to determine whether the attempt is successful:

```
TBool COandXAppView::TryHitSquareL(const COandXTile* aControl)
  {
  return Controller().HitSquareL(Index(aControl));
  }
```

It returns `ETrue` only if the square was previously empty and now contains a symbol, otherwise it returns `EFalse`.

As was mentioned earlier, `TileStatus()` is called from each tile when it needs to draw itself, in order to determine which, if any, symbol should be drawn in the tile. The function's action is simply to request the information from the engine:

```
TTileState COandXAppView::TileStatus (const COandXTile* aControl) const
  {
  return Engine().TileStatus(Index(aControl));
  }
```

The majority of the view's functions are described in Chapter 15, where they are used to illustrate the general behavior of controls. Here, we mention only `ShowTurn()`, which is called from the application UI class and the controller whenever there is a change in whose turn it is to play. It simply requests the status window to redraw itself:

```
void COandXAppView::ShowTurn()
  {
  iStatusWin->DrawDeferred();
  }
```

How `DrawDeferred()` works is explained in Chapter 17; here it is sufficient to know that it results in a call to the status window's `Draw()` function. As we saw earlier, there is no need for `ShowTurn()` to pass a parameter indicating whose turn it is, since the status window's `Draw()` function queries the controller directly for the symbol it needs to draw.

Command Menus

The application uses only two commands: one to specify whether the Nought player or the Cross player should make the first move, and one

FIFTH THIRD EX1062
PART 2 of 3
Page 43 of 379

to abandon the current game. The first of these is only relevant before a game starts and the second is only used while a game is in progress. In consequence, we've chosen to use a dynamic command menu, which shows only those commands that can sensibly be used, given the current state of the game. We've implemented the menu in a way that illustrates some of the options that are available for setting up and changing the content of a command menu. Although at least one of the commands, that to select the first player, could reasonably be implemented via a dialog, we've chosen not to do so, leaving the discussion of dialog use to Chapter 16.

For S60, the specification of the command menu starts in the resource script, with the definition of an `EIK_APP_INFO` resource:

```
RESOURCE EIK_APP_INFO
  {
  menubar = r_oandx_menubar;
  cba = R_AVKON_SOFTKEYS_OPTIONS_EXIT;
  }
```

This resource contains a pointer to the resource that contains the application's menu bar and a Control Button Array that defines the softkey labels – in this case, Options and Exit – that are to appear in the control pane.

The menu bar resource, in turn, specifies one or more `MENU_PANE` resources that list the menu items they contain. As in this case, an S60 menu bar normally contains only one menu pane. If you specify more than one menu pane, the S60 UI concatenates their contents into a single list.

```
RESOURCE MENU_BAR r_oandx_menubar
  {
  titles =
    {
    MENU_TITLE
      {
      menu_pane = r_oandx_menu;
      }
    };
  }
```

For this application, the menu pane specifies only one menu item; the second menu item is constructed dynamically.

```
RESOURCE MENU_PANE r_oandx_menu
  {
  items =
    {
    MENU_ITEM
```

```
    {
    command = EOandXNewGame;
    txt = "New game";
    }
  };
}
```

Each menu item specifies the text to be displayed and a non-zero value, unique within the application, used to identify the command. The command IDs are normally defined via an enumeration in the application's HRH file; for this application, they are defined, in `oandx.hrh`, as:

```
enum TOandXIds
  {
  EOandXNewGame = 0x1000,  // start value must not be 0
  EOandXSwitchTurn
  };
```

The resource file specifies only the command ID and text for the New game command. The other command is added dynamically according to context, using one or other of the following two resources for its text:

```
RESOURCE TBUF r_oandx_o_moves_first
  {
  buf = "Noughts move first";
  }

RESOURCE TBUF r_oandx_x_moves_first
  {
  buf = "Crosses move first";
  }
```

It is worth noting here that a truly commercial application would read the text for these resources – and most other text within the resource script – from a separate file, in order to facilitate translation into other languages. We have not done so in order to make the meaning of these resources clearer in the text of this book.

S60 applications that need to modify the content of their command menus should supply an implementation of the `DynInitMenuPaneL()` function of the application UI class, which is called each time a menu is about to be displayed. The `aResourceId` parameter identifies the menu and `aMenuPane` points to a menu pane that has already been constructed from the relevant resource. For this application, the implementation is:

```
void COandXAppUi::DynInitMenuPaneL(TInt aResourceId,
                                   CEikMenuPane* aMenuPane)
  {
  if (aResourceId == R_OANDX_MENU)
```

```
    {
    if (iController->IsNewGame())
      {
      CEikMenuPaneItem::SData item;
      iCoeEnv->ReadResource(item.iText,
          iController-IsCrossTurn() ? R_OANDX_O_MOVES_FIRST :
                                      R_OANDX_X_MOVES_FIRST);
      item.iCommandId = EOandXSwitchTurn;
      item.iFlags = 0;
      item.iCascadeId = 0;
      aMenuPane->AddMenuItemL(item);

      aMenuPane->DeleteMenuItem(EOandXNewGame);
      }
    }
  }
```

The function first checks that the resource ID is for the relevant menu, but does not modify the menu pane (which therefore displays the New game menu item) if a game is in progress. Otherwise, a new menu item is constructed, in a `CEikMenuPaneItem::SData` data structure, using the relevant resource text, and added to the menu pane, and the New game item is deleted.

When the user selects a command, the S60 UI responds by calling the `HandleCommandL()` function of the application UI class. This is normally implemented via a `switch` statement, with each case corresponding to one of the possible command IDs:

```
void COandXAppUi::HandleCommandL(TInt aCommand)
  {
  switch(aCommand)
    {
    case EEikCmdExit:
    case EAknSoftkeyExit:
      {
      SaveL();
      Exit();
      }
      break;
    case EOandXNewGame:
      {
      iController->Reset();
      iAppView->ResetView();
      }
      break;
    case EOandXSwitchTurn:
      {
      iController->SwitchTurn();
      }
      break;
    default:
      break;
    }
  }
```

FIFTH THIRD EX1062
PART 2 of 3
Page 46 of 379

In the Noughts and Crosses application, each case, with the exception of the Exit command, corresponds to one of the command IDs specified in `oandx.hrh` and is handled simply by making the appropriate changes to the controller's state.

Depending on how the Exit command is initiated, it may have either of the standard command IDs shown in the above code. The application's response must always be to call `Exit()` but, in addition, it calls `SaveL()` to save the current game state in the application's persistent data.

12.2 Differences for UIQ 3

One of the major influences in the development of UIQ 3 was the need to make it easier for developers to support multiple form factors (both in terms of screen dimensions and use of portrait or landscape alignments) and a variety of input methods (touch screen or keyboard; menu-driven or softkey-driven interaction) from a single code base.

In order to facilitate such flexibility, all UIQ 3 applications need to derive their views from the new `CQikViewBase` or `CQikMultiPageViewBase` classes. The UIQ version of the Noughts and Crosses application derives its main view from the `CQikViewBase` view class (which is a subclass of `CCoeControl`) and not directly from `CCoeControl` itself. However, since it only has one view, it makes no specific use of the view architecture that is described in Chapter 14.

The Application UI Class

There is a small difference in the way that the application UI class creates and initializes the View. Remember that the S60 `ConstructL()` function of the application UI class is implemented as:

```
void COandXAppUi::ConstructL()
  {
  BaseConstructL(EAknEnableSkin);
  iEngine = COandXEngine::NewL();
  iController = COandXController::NewL();
  iAppView = COandXAppView::NewL(ClientRect());
  AddToStackL(iAppView); // Enable keypresses to the view
  iStacked = ETrue;
  ReportWhoseTurn();
  }
```

The destructor needed to take note of `iStacked` in order to determine whether or not it should remove the view from the control stack. For UIQ, the corresponding code is:

```
void COandXAppUi::ConstructL()
  {
```

FIFTH THIRD EX1062
PART 2 of 3
Page 47 of 379

```
CQikAppUi::ConstructL(); // initiate the standard values
iEngine = COandXEngine::NewL();
iController = COandXController::NewL();
// Create the view and add it to the framework
iAppView = COandXView::NewL(*this);
AddViewL(*iAppView);

ReportWhoseTurn();
}
```

The most significant difference is that the UIQ version registers the view by calling `AddViewL()`, which also takes care of adding it to the control stack. After this call, the application UI class has full ownership of the view, so you do not have to worry about either deregistering the view or removing it from the control stack when the application UI class is destroyed.

The View Class

The majority of the remaining source code differences that are needed in a UIQ version of the application stem from the differences between `CQikViewBase` and `CCoeControl`. So, while both forms of the view own their component controls, the mechanisms of ownership and access are different. A view that derives from `CCoeControl`, as in the S60 version, creates its component controls (normally in its `ConstructL()` function) and stores pointers to them in specific items of member data. In the S60 version, the relevant `ConstructL()` code is:

```
...
for (TInt i = 0; i < KNumberOfTiles; i++)
  {
  User::LeaveIfError(iTiles.Append(CreateTileL()));
  }
...
iStatusWin = COandXStatusWin::NewL(Window());
```

The controls are normally explicitly deleted in the destructor:

```
COandXAppView::~COandXAppView()
  {
  for (TInt i=0; i<KNumberOfTiles; i++)
    {
    delete iTiles[i];
    }
  iTiles.Close();
  delete iStatusWin;
  }
```

As described more fully in Chapter 17, access to each child control is via the `CountComponentControls()` and `ComponentControl()` functions that each component-owning control must implement:

```
TInt COandXAppView::CountComponentControls() const
  {
  return KNumberOfTiles + 1;
  }

CCoeControl* COandXAppView::ComponentControl(TInt aIndex) const
  {
  if (aIndex==KNumberOfTiles)
    {
    return iStatusWin;
    }
  else
    {
    return const_cast<COandXTile*>(iTiles[aIndex]);
    }
  }
```

`CQikViewBase` provides its own mechanism for component control ownership, so views that derive from this class do not need to provide explicit data members to store pointers to the components, nor do they need to provide code in the destructor to delete them. In addition, such views do not need to (and should not) provide implementations of `CountComponentControls()` and `ComponentControl()`.

Instead, UIQ views add component controls – normally from within the view's implementation of `ViewConstructL()` – to a built-in array that is initialized by a call to `InitComponentArrayL()` and then accessed via a `Components()` function, for example:

```
void COandXView::ViewConstructL()
  {
  ViewConstructFromResourceL(R_OANDX_UI_CONFIGURATIONS);
  InitComponentArrayL();
  for (TInt i = 0; i < KNumberOfTiles; i++)
    {
    COandXTile * tile = new(ELeave) COandXTile;
    AddControlLC(tile, i);
    tile->ConstructL(this);
    tile->SetFocusing(ETrue);
    CleanupStack::Pop(tile);
    }
  //  Status window added, but not created here
  AddControlLC(iStatusWin, KNumberOfTiles);
  CleanupStack::Pop(iStatusWin);
  iStatWinNotAppended = EFalse;
  // Status window now owned by the view

  UpdateCommandsL(Controller().IsNewGame(), Controller().IsCrossTurn());
  SetFocusByIdL(0);
  }
```

In the UIQ version of the application, as you can see from the above code, each tile is created and then appended to the array with a call to `AppendLC()` which, as its name implies, leaves a pointer to the

tile on the cleanup stack. The tile is removed from the cleanup stack once any potentially leaving initialization is complete. The view takes ownership of all component controls that are added in this way and they are automatically deleted when the view is destroyed.

> If you think carefully about the code sequence for adding a component into a view, you might begin to wonder if it is truly safe. For example, what if the actions performed by `AddControlLC()` cause a leave before the control is added to the cleanup stack? Furthermore, if the view takes ownership of the control, should `AddControlLC()` leave the control on the cleanup stack at all? Could a leave before the control is popped from the cleanup stack result in the control being deleted twice – once from the cleanup stack and once by the owning view? In Symbian OS, it is worth asking questions like this, since leave-safeness is paramount.
>
> In fact, this sequence is safe. `AddControlLC()` adds the control to the cleanup stack before performing any potentially leaving actions, and the item that is added to the cleanup stack is one that is specifically designed to transfer ownership of the view (in a leave-safe way) as it is popped.

The creation of the status window follows a slightly different pattern. The reason for this is that the status window is accessed from the controller, via the `ReportWhoseTurn()` function of the application UI class, and the first access occurs before the view's `ViewConstructL()` has been called. (To optimize start-up time, particularly for applications with multiple views, `ViewConstructL()` is only called at the time that the view first needs to be made visible.) To prevent this attempted access from causing a panic, we need to create the status window in the view's `ConstructL()` and add it in `ViewConstructL()`.

However, there is then a risk that the status window will not be deleted if the code leaves between its creation and it being successfully added. We solved this problem by including the `iStatWinNotAppended` flag in the class data, setting it when the status window is created and clearing it when the window has successfully been added to the array. The view's destructor has to delete the status window only if the flag is set:

```
COandXView::~COandXView()
  {
  if (iStatWinNotAppended)
  // Once appended, system code takes care of a component's deletion
    {
    delete iStatusWin;
    }
  }
```

FIFTH THIRD EX1062
PART 2 of 3
Page 50 of 379

It is also worth pointing out that the initial construction of the view, from `ViewConstructL()`, is by means of a call to `ViewConstructFromResourceL()`. In suitable cases, an entire multi-page view can be constructed from the resource accessed by this function, in the same way that a dialog can be constructed from a resource, as described in Chapter 16. In the present case, the resource specifies a single page, without supplying any content:

```
RESOURCE QIK_VIEW_PAGES r_oandx_layout_pages
  {
  pages =
    {
    QIK_VIEW_PAGE
      {
      page_id = EOandXViewPage;
      }
    };
  }
```

Since the UIQ version of the view class does not have accessible implementations of `ComponentControl()` or `CountComponentControls()`, functions that, in an S60 application, call one or both of these functions need a different implementation. In the Noughts and Crosses application, the functions that are affected are those associated with which tile has focus. In the S60 version, these are `IdOfFocusControl()` and `MoveFocusTo()`:

```
TInt COandXAppView::IdOfFocusControl()
  {
  TInt ret = -1;
  for (TInt i=0; i<KNumberOfTiles; i++)
    {
    if (ComponentControl(i)->IsFocused())
      {
      ret = i;
      break;
      }
    }
  ASSERT_ALWAYS(ret>=0, Panic(EOandXNoTileWithFocus));
  return ret;
  }

void COandXAppView::MoveFocusTo(const TInt index)
  {
  TInt oldIndex = IdOfFocusControl();
    if (index != oldIndex)
    {
    SwitchFocus(oldIndex, ComponentControl(index));
    }
  }
```

`SwitchFocus()` is implemented as:

```
void COandXAppView::SwitchFocus(TInt aFromIndex, CCoeControl* aToControl)
  {
  ComponentControl(aFromIndex)->SetFocus(EFalse, EDrawNow);
  aToControl->SetFocus(ETrue, EDrawNow);
  }
```

In the UIQ version, the two corresponding functions are `IdOfFocusedControl()` and `SetFocusByIdL()`, whose basic implementations are:

```
TInt COandXView::IdOfFocusedControl()
  {
  TInt i;
  for (i=0; i<KNumberOfTiles; i++)
    {
    CCoeControl *control=ControlById<CCoeControl>(i);
    if (control->IsFocused())
      break;
    }
  return i;
  }

void COandXView::SetFocusByIdL(TInt aControlId)
  {
  RequestFocusL(ControlById<CCoeControl>(aControlId));
  }
```

In the UIQ version, we changed the names of both functions to emphasize the differences in implementation. Note, in particular, that `SetFocusByIdL()` is a leaving function, whereas `MoveFocusTo()` is not. Fortunately this does not give rise to any other complications in the current application, since they are called only from within leaving functions.

While on the subject of focus, there are two further differences that need mentioning. First, the default status of newly created controls in S60 is for them to be capable of accepting focus, which is not the case in UIQ. In UIQ you need to call `SetFocusing(ETrue)` on each control that needs to accept focus (in this case, on each of the board's tiles). In contrast, in S60, you need to call `SetFocusing(EFalse)` on each control, such as the game's status window, that you do not want to be able to accept focus. Secondly, UIQ 3 views provide automatic, intelligent key- or button-based navigation between any of the component controls that can accept focus – that is, those controls for which `SetFocusing(ETrue)` has been called. In consequence, the UIQ version has no need to implement the view's `OfferKeyEventL()` function.

Commands

UIQ 3 views are primarily designed for use in applications with multiple views, each with its own set of commands, so they are not seen at their

FIFTH THIRD EX1062
PART 2 of 3
Page 52 of 379

best in an application such as the Noughts and Crosses application, which only has a single view. The application does, however, allow the basic differences to be illustrated.

A major difference in UIQ is that commands are treated in a more abstract way than in S60. A UIQ device may be set up to access its commands by means of either a pointer device or softkeys, and the commands themselves may or may not be displayed in a command menu. In addition, commands may be provided from a variety of sources within the application, such as one or more individual controls – or even from outside the application itself, for example, from a front-end processor.

The UIQ application framework contains a command model that deals with all these variations. In consequence, the application programmer need only be concerned with which commands should be available and not with how they are presented or selected.

The first change is to the resource script. As we saw earlier, an S60 application uses an `EIK_APP_INFO` resource, to specify the menu bar and the menu items it contains. In contrast, a UIQ 3 application must specify an empty `EIK_APP_INFO` resource, plus a `QIK_VIEW_CONFIGURATIONS` resource that lists the UI configurations it supports. In the Noughts and Crosses application, these resources are:

```
RESOURCE EIK_APP_INFO { }

RESOURCE QIK_VIEW_CONFIGURATIONS r_oandx_ui_configurations
  {
  configurations =
    {
    QIK_VIEW_CONFIGURATION
      {
      ui_config_mode = KQikPenStyleTouchPortrait;
      command_list = r_oandx_portrait_commands;
      view = r_oandx_layout;
      },
    QIK_VIEW_CONFIGURATION
    {
      ui_config_mode = KQikSoftkeyStylePortrait;
      command_list = r_oandx_portrait_commands;
      view = r_oandx_layout;
      }
    };
  }
```

Each configuration specifies its list of available commands via a `QIK_COMMAND_LIST` resource:

```
RESOURCE QIK_COMMAND_LIST r_oandx_portrait_commands
  {
  items =
    {
```

```
    QIK_COMMAND
      {
      id = EEikCmdExit;
      type = EQikCommandTypeScreen; // Only visible in debug
      stateFlags = EQikCmdFlagDebugOnly;
      text = "Close (debug)";
      },
    QIK_COMMAND
      {
      id = EOandXNewGame;
      type = EQikCommandTypeScreen;
      text = "New game";
      },
    QIK_COMMAND
      {
      id = EOandXFirstPlayer;
      type = EQikCommandTypeScreen;
      text = "Noughts move first";
      }
    };
  }
```

As in the S60 version, the commands are updated dynamically, but the mechanism is different. S60 applications update their menus via the `DynInitMenuPaneL()` function, which is called immediately before a menu is made visible.

In UIQ, since a command may be selected by means of a softkey, or from a toolbar, there is no guarantee that a menu pane is displayed before the selection takes place. In consequence, you cannot use `DynInitMenuPaneL()` to update the available commands. Indeed, since some commands may be permanently visible, UIQ applications need to perform such updating each time the view's state changes in a way that affects the commands.

In many cases, it would be appropriate to implement the view's `HandleControlEventL()` function, which is called (with an `EEventStateChanged` event code) each time one of the view's controls changes its state. In the Noughts and Crosses application, all significant state changes take place in the Controller, so we've taken the more direct option of adding a `SetStateL()` function to the Controller, implemented as:

```
void COandXController::SetStateL(TState aState)
  {
  iState = aState;
  OandXAppUi()->UpdateCommandsL(IsNewGame(), IsCrossTurn());
  }
```

This calls an `UpdateCommandsL()` function in the application UI class, which simply calls a similarly named function in the view:

```
void COandXView::UpdateCommandsL(TBool aIsNew, TBool aIsCrossTurn)
  {
```

FIFTH THIRD EX1062
PART 2 of 3
Page 54 of 379

```
iCommandManager.SetAvailable(*this, EOandXNewGame, !aIsNew);
iCommandManager.SetAvailable(*this, EOandXFirstPlayer, aIsNew);
iCommandManager.SetTextL(*this, EOandXFirstPlayer,
    aIsCrossTurn?R_OANDX_O_FIRST:R_OANDX_X_FIRST);
}
```

The following points are relevant when updating the menus:

- While a game is in progress, there is no need to worry about whether the text for the 'Who moves first' command is updated correctly or not, because that command is only available (and visible) before a new game is started.
- When executing the 'Who moves first' command, the game is always new, so the only interest is in setting the correct text for the next potential use of that command.

In addition, the controller's `Reset()` function is renamed to `ResetL()` because it can now leave:

```
void COandXController::ResetL()
  {
  Engine().Reset();
  if (IsCrossTurn())
    {
    SwitchTurn();
    }
  SetStateL(ENewGame);
  }
```

`ResetL()` cannot be used to initialize the controller (and engine) because `SetStateL()` calls the view's `UpdateCommandsL()` function, and the view does not exist at the time the controller is constructed. Instead, in the controller's `ConstructL()`, we explicitly set the relevant data:

```
...
Engine().Reset();
iState = ENewGame;
iCrossTurn = EFalse;
...
```

Since UIQ expects an application to have multiple views and each view may have a different set of commands, command processing is implemented within each individual view. A UIQ application, therefore, does not supply a `HandleCommandL()` function in the application UI class, but implements this function in each of its views. Note that the argument to this function is a reference to an instance of `CQikCommand`, rather than the command ID that is passed to this function in the

application UI class. Otherwise, the implementation is broadly similar to that in the S60 version:

```
void COandXView::HandleCommandL(CQikCommand& aCommand)
  {
  switch(aCommand.Id())
    {
    case EOandXNewGame:
      Controller().ResetL();
      SetFocusByIdL(0);
      DrawNow();
      break;
    case EOandXFirstPlayer:
      Controller().SwitchTurn();
      UpdateCommandsL(Controller().IsNewGame(),
                      Controller().IsCrossTurn());
      break;
    default:
#ifdef _DEBUG
      OandXAppUi()->SaveGameStateL();
#endif
      // The Go back and Exit commands are handled by CQikViewBase.
      CQikViewBase::HandleCommandL(aCommand);
      break;
      }
  }
```

Persistence

In UIQ, the default behavior is to allow applications to use persistent data, so, unlike in S60, there is no need to supply an implementation of the document's `OpenFileL()` function.

UIQ applications do not normally have a Close command (except for testing purposes, in debug builds). Instead, system code may instruct an application to close at any time that memory needs to be freed, first giving it an opportunity to save its persistent data. In order to access the functionality more easily, we have added – but only in debug builds – a call to the `SaveGameStateL()` function of the application UI class in the default case of the view's `HandleCommandL()`. The implementation is very simple:

```
void COandXAppUi::SaveGameStateL()
  {
  SaveL();
  }
```

We have to supply this new function because the `SaveL()` function of the application UI class is protected, and so cannot be called directly from the view. With this call in place, the persistent data is saved each time the Close (Debug) command is selected.

FIFTH THIRD EX1062
PART 2 of 3
Page 56 of 379

Since the application's persistent data contains one item – the ID of the tile that currently has focus – from the view, there is one final complication that we have to deal with. It is caused by the view's controls being created and initialized in the `ViewConstructL()` function which, as mentioned earlier, is called as late as possible, just before the view first becomes visible. This means that the document's `RestoreL()` function (and, at least when the application is run for the first time, its `StoreL()` function) are called before the view is fully initialized.

We resolve this problem by adding an `iInitialFocusId` data member to the `COandXView` class and coding the view's `SetFocusByIdL()` function as:

```
void COandXView::SetFocusByIdL(TInt aControlId)
  {
  CCoeControl * control = ControlById<CCoeControl>(aControlId);
  if (control)
    {
    RequestFocusL(control);
    }
  else // Control does not yet exist
    {
    iInitialFocusId = aControlId;
    }
  }
```

If the control does not yet exist, the value is copied into `iInitalFocusId` and used, within `ViewConstructL()`, to move focus to the correct control after it has been created.

We also need to deal with the case where the persistent data needs to be written before the tiles exist (this only happens once, the first time the application is run, when the application's INI file is being created). The relevant data is obtained by a call to the view's `IdOfFocusedControl()`, whose implementation is:

```
TInt COandXView::IdOfFocusedControl()
  {
  TInt i;
  for (i=0; i<KNumberOfTiles; i++)
    {
    CCoeControl *control=ControlById<CCoeControl>(i);
    if (!control) // Not yet created
      {
      return 0;
      }
    if (control->IsFocused())
      {
      break;
      }
    }
  return i;
  }
```

If a control does not yet exist, the function simply returns zero, which is the correct default value.

Summary

This chapter has described a simple but non-trivial application to play a game of Noughts and Crosses.

After a brief introduction to the game, the chapter explained the overall architecture of the application, before describing each of the S60 version's major component classes in more detail.

The chapter concluded with a discussion of the main differences between the S60 and UIQ versions of the game.

FIFTH THIRD EX1062
PART 2 of 3
Page 58 of 379

21

Multimedia

This chapter describes the Multimedia Framework and the associated Multimedia APIs that are present in Symbian OS v9.1. These APIs can be used to play and record audio and video data, perform image processing and manipulation, and access any camera and radio tuner hardware present on a phone.

21.1 The Multimedia Framework

The Multimedia Framework (MMF) is a framework, based on ECOM plug-ins, that is used for playing back and recording audio and video clips. It allows phone manufacturers and third parties to add plug-ins to provide support for more audio and video formats. For the application programmer, it provides APIs that abstract away from the underlying hardware, thereby simplifying the code needed to record and play the supported formats. The streaming APIs, which bypass large parts of the MMF, provide a lower-level interface to allow streaming of audio data to and from the audio hardware.

The MMF is a lightweight framework that makes use of a worker thread to process the audio or video data. It makes extensive use of ECOM to allow for extensibility via plug-ins. Figure 21.1 shows its basic structure during audio- or video-clip playback or recording.

Applications use the client APIs to control the playback or recording of media. The underlying controller framework creates a new thread to process the media data and uses the Symbian client–server framework to communicate with this thread. The audio streaming APIs and the audio tone API interface directly to the MDA layer; they do not use a separate thread.

FIFTH THIRD EX1062
PART 2 of 3
Page 59 of 379

Appendix

Developer Resources

Download the source code for this book at
developer.symbian.com/main/academy/press/books/scmp_v3.

Symbian OS Software Development Kits

Software development kits (SDKs) are based on a particular user interface (UI) for Symbian OS. Each distinct UI has an associated set of system applications for such tasks as messaging, browsing, telephony, multimedia and contact and calendar management. These applications typically make use of generic application engines provided by Symbian OS. SDKs are required to support the installation of third-party applications written in native C++. There are two UIs for Symbian OS v9, SDKs for which can be found at the websites listed here:

- Nokia S60 (***www.forum.nokia.com/S60***)
- UIQ (***developer.uiq.com/devtools_uiqsdk.html***)

For the independent software developer, the most important thing to know in targeting a particular phone is its associated reference platform. You can then decide which SDK you need to obtain. In most cases, you can target – with a single version of your application – all phones based on the same reference platform and Symbian OS version associated with this SDK. The Symbian OS System Definition papers give further details of possible differences between phones based on a single SDK (***www.symbian.com/phones***).

Getting an IDE for Development

To develop your code, you'll need an Integrated Development Environment (IDE), which contains an editor, compiler and linker. Two IDEs

FIFTH THIRD EX1062
PART 2 of 3
Page 60 of 379

are recommended – for non-commercial development, use Carbide.c++ Express Edition, which is a free download. Otherwise, Carbide.c++ Developer and Professional Editions are recommended, along with CodeWarrior. Support for Visual Studio .NET 2003 is also available using Carbide.vs.

- Carbide.c++ (***www.forum.nokia.com/carbide***)
- CodeWarrior (***www.forum.nokia.com/codewarrior***)

Getting a UID for Your Application

A UID is a 32-bit number. Every graphical application should have its own UID that allows Symbian OS to distinguish files associated with that application from files associated with other applications. UIDs are also used in other circumstances, such as to identify streams within a store and to identify one or more of an application's views.

To get a UID, register with and log on to the Symbian Signed website (***www.symbiansigned.com***). Request your UIDs from the My Symbian Signed tab – ten UIDs is a reasonable first request.

> Don't build different Symbian OS applications with the same application UID – even the same test UID – on your emulator or Symbian OS machine. If you do, the system recognizes only one of them and you will not be able to launch any of the others.

Symbian OS Developer Tools

As well as tools offerings from Symbian partners, Symbian Developer Network (***developer.symbian.com/main/tools***) provides a number of free, unsupported and open source tools.

Support Forums

Symbian Developer Network offers newsgroups and support forums:

- ***news://developer.symbian.com***
- ***http://developer.symbian.com/forum***

Symbian Developer Network partners also offer support for developers.

FIFTH THIRD EX1062
PART 2 of 3
Page 61 of 379

Sony Ericsson Developer World

As well as tools and SDKs, Sony Ericsson Developer World (***developer.sonyericsson.com***) provides a range of services including newsletters and support packages for developers working with the latest Sony Ericsson products such as the P990.

Forum Nokia

As well as tools and SDKs, Forum Nokia (***forum.nokia.com/main.html***) provides newsletters, the Knowledge Network, fee-based case-solving, a Knowledge Base of resolved support cases, discussion archives and a wide range of C++ and Java-based technical papers of relevance to developers targeting Symbian OS.

Sun Microsystems Developer Services

In addition to providing a range of tools and SDKs, Sun also provides a wide variety of developer support services including free forums, newsletters, and a choice of fee-based support programs.

- Forums: ***forum.java.sun.com***
- Support and newsletters: ***developer.java.sun.com/subscription***

Symbian OS Developer Training

Symbian's Technical Training team and Training Partners offer public and on-site developer courses around the globe. Course dates and availability are listed at ***www.symbian.com/developer/training***.

Symbian normally offers a 20 % discount on all bookings confirmed up to one month before the start of any course. This discount cannot be used in conjunction with any other discounts.

Course	Level	Language
Symbian OS essentials	Introductory	C++
Java on Symbian OS	Introductory	Java
Symbian OS: Application engine development	Intermediate	C++
Symbian OS: Application UI development	Intermediate	C++
Symbian OS: Internals	Advanced	C++
Symbian OS: UI system creation	Advanced	C++

FIFTH THIRD EX1062
PART 2 of 3
Page 62 of 379

Intermediate and advanced courses require previous attendance at the OS Essentials course. The UI system creation course also requires previous attendance at the Application UI course.

Developer Community

These community websites offer news, reviews, features and forums, and represent a rich alternative source of information that complements the Symbian Developer Network and the development tools publishers. They are good places to keep abreast of new software and to announce the latest releases of your own applications.

My-Symbian

My-Symbian (***my-symbian.com***) is a website based in Poland and dedicated to news and information about Symbian OS phones. This site presents descriptions of new software for Symbian OS, classified by user interface. It also features discussion forums and an online shop.

All About Symbian

All About Symbian (***www.allaboutsymbian.com***) is a website based in the UK and dedicated to news and information about Symbian OS phones. The site features news, reviews, software directories and discussion forums. It has strong OPL coverage.

SymbianOne

SymbianOne (***www.symbianone.com***) features news, in-depth articles, case studies, employment opportunities and event information all focused on Symbian OS. A weekly newsletter provides up-to-date coverage of developments affecting the Symbian OS ecosystem. This initiative is a joint venture with offices in Canada and New Zealand.

NewLC

NewLC (***www.newlc.com***) is a collaborative website based in France and dedicated to Symbian OS C++ development. It aims to be initially valuable to developers just starting writing C++ applications for Symbian OS; in time it will cover more advanced topics.

infoSync World

infoSync World (***symbian.infosyncworld.com***) is a website based in Norway that provides features, news, reviews, comments and a wealth

FIFTH THIRD EX1062
PART 2 of 3
Page 63 of 379

of other content related to mobile information devices. It features a section dedicated to Symbian OS, covering new phones, software and services – mixed with strong opinions that infoSync is not afraid to share.

TodoSymbian

TodoSymbian (***www.todosymbian.com***) is a website written in Spanish. It provides news, reviews, software directories, discussion forums, tutorials and a developers' section.

Symbian OS Books

Development

Symbian OS C++ for Mobile Phones, Vol. 1, Harrison *et al.* John Wiley & Sons. ISBN: 0470856114

Symbian OS C++ for Mobile Phones, Vol. 2, Harrison *et al.* John Wiley & Sons. ISBN: 0470871083

Symbian OS Explained, Stichbury. John Wiley & Sons. ISBN: 0470021306

Developing Software for Symbian OS, Babin. John Wiley & Sons. ISBN: 0470018453

Symbian OS Internals, Sales *et al.* John Wiley & Sons. ISBN: 0470025247

Symbian OS Platform Security, Heath *et al.* John Wiley & Sons. ISBN: 0470018828

Symbian OS C++ Architecture, Morris. John Wiley & Sons. ISBN: 0470018461

Programming PC Connectivity Applications for Symbian OS, McDowall. John Wiley & Sons. ISBN: 0470090537

S60 Programming – A Tutorial Guide, Coulton *et al.* John Wiley & Sons. ISBN: 9780470027653

Management

Symbian for Software Leaders, Wood. John Wiley & Sons. ISBN: 0470016833

Symbian Academy

The Accredited Symbian Developer Primer, Stichbury and Jacobs. John Wiley & Sons. ISBN: 0470058277

Smartphone Operating System Concepts with Symbian OS, Jipping. John Wiley & Sons. ISBN: 0470034491

FIFTH THIRD EX1062
PART 2 of 3
Page 64 of 379

APPENDIX
HARRISON02

Book

Symbian OS C++ for mobile phones / Richard Harrison with Alan Robinson ... [et al.] ; reviewed by Andrew Thoelke ... [et al.].

By: Harrison, Richard
Contributors: Shackman, Mark
Published: Wiley, Chichester, West Sussex, England, c2003-<c2007>



Ask a librarian

Available: 2 of 3 volumes remaining

Request item | Item availability

Full record | **MARC record**

Staff view

Download

LEADER	02217cam a22004574a 4500
001	13126294
005	20250606200346.8
008	030317m20039999enka 001 0 eng
035	$a 13126294
906	$a 7 $b cbc $c orignew $d 1 $e ecip $f 20 $g y-gencatlg
925 0	$a acquire $b 2 shelf copies $x policy default
955	$a jg12 2003-03-17 $i jg12 2003-03-17 $e jg07 2003-03-18 to Dewey $a aa07 2003-03-18 $a ps16 2003-06-18 bk rec'd, to CIP ver. $a jf00 2003-07-29 $f px06 2003-09-23 CIP ver. to BCCD $a ja15 2004-09-14 Copy 2 to BCCD $c jf07 2007-10-31 opened and added v. 3
955	$a ADDED VOLS: v. 3 ps02 2007-10-16 to ASCD
955	$a LCAP batch update 2025-06-01-04:00: LCAPM-644
010	$a 2003006223
020	$a 0470856114 $q Paper : alk. paper
020	$a 9780470066416 $q v. 3
040	$a DLC $c DLC $d DLC
042	$a pcc
050 0 0	$a TK6570.M6 $b H295 2003
082 0 0	$a 621.3845/6 $2 21
100 1	$a Harrison, Richard.
245 1 0	$a Symbian OS C++ for mobile phones / $c Richard Harrison with Alan Robinson ... [et al.] ; reviewed by Andrew Thoelke ... [et al.].

Ask a librarian
Ask a librarian

260	$a Chichester, West Sussex, England : $b Wiley, $c c2003-<c2007>
300	$a v. <1, 3> : $b ill. ; $c 24 cm. + $e 1 CD-ROM (4 3/4 in.)
336	$a text $b txt $2 rdacontent
337	$a unmediated $b n $2 rdamedia
338	$a volume $b nc $2 rdacarrier
500	$a Vol. 3 has authors: Richard Harrison, Mark Shackman with ... [et al.] ; reviewed by Graeme Duncan ... [et al.].
500	$a Vol. 1 does not have a volume number; includes 1 CD-ROM (4 3/4 in.)
500	$a Vol. 3 does not have a CD-ROM.
500	$a Includes indexes.
630 0 0	$a Symbian OS (Computer file)
650 0	$a Cell phone systems $x Computer programs.
650 0	$a C++ (Computer program language)
650 0	$a Operating systems (Computers)
700 1	$a Shackman, Mark.
856 4 2	$3 Contributor biographical information $u http://www.loc.gov/catdir/bios/wiley046/2003006223.html
856 4 2	$3 Publisher description $u http://www.loc.gov/catdir/description/wiley039/2003006223.html
856 4 1	$3 Table of contents $u http://www.loc.gov/catdir/toc/wiley031/2003006223.html
999 f f	$i 3ebaf99d-1a11-5a59-be14-0db7a64c4f2f $s 8e79c93c-6726-44a1-be51-1758c0cd4bb8

Please contact library staff for additional help

Availability & locations

Items: **1-3** of **3**

Status	Location	Call number
In Transit	General Collections Onsite	TK6570.M6 H295 2003 • (v. 3)
Available	Machine Readable Collections - Offsite (Jefferson, LJ139) Offsite	TK6570.M6 H295 2003 FT MEADE
Available	Machine Readable Collections - Offsite (Jefferson, LJ139) Offsite	TK6570.M6 H295 2003 FT MEADE

Ask a librarian | Feedback | Contact us

Speech Enabled | Accessibility | Legal | Inspector general | External link disclaimer | USA.gov

EBSCO Help | Privacy policy | Terms of use

Software © 2026 EBSCO Industries, LLC. All rights reserved

Version 2.18.5.v2.18.5_2026_06_14_19_04_04

Book

Symbian OS C++ for mobile phones / Richard Harrison with Alan Robinson ... [et al.] ; reviewed by Andrew Thoelke ... [et al.].



By: Harrison, Richard

Contributors: Shackman, Mark

Published: Wiley, Chichester, West Sussex, England, c2003-<c2007>

Available: 2 of 3 volumes remaining

Request item | Item availability

Full record | MARC record

Additional links

Permalink

https://lccn.loc.gov/2003006223

Links

- Contributor biographical information
- Publisher description
- Table of contents

Ask a librarian

Publication details

Language
English

Publisher
Wiley, Chichester, West Sussex, England

Library of Congress subjects
Symbian OS (Computer file)
Cell phone systems--Computer programs
C++ (Computer program language)
Operating systems (Computers)

Document type
Book

Physical description
v. <1, 3> : ill. ; 24 cm. + 1 CD-ROM (4 3/4 in.)

Ask a librarian

Contents/Summary

Notes
Vol. 3 has authors: Richard Harrison, Mark Shackman with ... [et al.] ; reviewed by Graeme Duncan ... [et al.]

Vol. 1 does not have a volume number; includes 1 CD-ROM (4 3/4 in.)

Vol. 3 does not have a CD-ROM...

Show more

Identifiers and numbering

ISBN
9780470066416
0470856114

LCCN
2003006223

Classification - LC
TK6570.M6 H295 2003

Classification - Dewey
621.38456

Please contact library staff for additional help

Availability & locations

Items: **1-3** of **3**

Status	Location	Call number
In Transit	General Collections Onsite	TK6570.M6 H295 2003 • (v. 3)
Available	Machine Readable Collections - Offsite (Jefferson, LJ139) Offsite	TK6570.M6 H295 2003 FT MEADE
Available	Machine Readable Collections - Offsite (Jefferson, LJ139) Offsite	TK6570.M6 H295 2003 FT MEADE

LIBRARY

Ask a librarian | Feedback | Contact us

Speech Enabled | Accessibility | Legal | Inspector general | External link disclaimer | USA.gov

EBSCO Help | Privacy policy | Terms of use

Software © 2026 EBSCO Industries, LLC. All rights reserved

Version 2.18.5.v2.18.5_2026_06_14_19_04_04

APPENDIX
HARRISON03

MARC View

LEADER 01656nam a2200361 a 4500				
001 3944836				
003 SIRSI				
005 20151215122100.0				
008 061107t20072007enka b 001 0 eng				
015				a| GBA696235 2| bnb
016	7			a| 013601779 2| Uk
019				a| MARS
020				a| 9780470066416 q| (pbk.)
020				a| 0470066415 q| (pbk.)
035				a| (OCoLC)76361852
040				a| UKM c| UKM d| BAKER d| BTCTA d| BWKUK d| YDXCP d| UtOrBLW
050		4		a| TK6570.M6 b| H295 2007
082	0	4		a| 621.38456028552768 2| 22
100	1			a| Harrison, Richard, d| 1946-
245	1	0		a| Symbian OS C++ for mobile phones. n| Vol. 3 / c| Richard Harrison, Mark Shackman ; with Adi Rome [and others].
264		1		a| Chichester, England ; a| Hoboken, NJ : b| Wiley, c| [2007]
264		4		c| ©2007
300				a| xxix, 803 pages : b| illustrations ; c| 24 cm
336				a| text b| txt 2| rdacontent
337				a| unmediated b| n 2| rdamedia
338				a| volume b| nc 2| rdacarrier
504				a| Includes bibliographical references and index.
583		1		a| committed to retain c| 20170930 d| 20421231 f| HathiTrust Shared Print Program u| https://www.hathitrust.org/shared_print_program 5| PSt
650		0		a| Cell phone systems x| Computer programs.
650		0		a| Operating systems (Computers)
650		0		a| C++ (Computer program language)
700	1			a| Shackman, Mark.
949				a| TK6570.M6H295 2007 w| LC c| 1 i| 000061022225 d| 5/31/2022 e| 3/21/2022 l| PATERNO-3 m| UP-PAT n| 2 r| Y s| Y t| BOOK u| 9/7/2007 o| .STAFF. Digitized as part of the HathiTrust & Google Books project o| .STAFF. This item is part of PSUL's commitment to the HathiTrust Shared Print Program. At least one physical copy must be retained in PSUL's holdings until January 1st, 2043.

ASK A LIBRARIAN

PennState

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

Legal Statements | PSU Hotlines

PENN STATE UNIVERSITY LIBRARIES

Libraries Home (814) 865-6368
Libraries Intranet (Staff Only)
Accessibility Help
Website Feedback
Policies and Guidelines
Staff Directory

CONNECT WITH PENN STATE UNIVERSITY LIBRARIES

Facebook
Twitter
Instagram



Symbian OS C++ for mobile phones. Vol. 3 / Richard Harrison, Mark Shackman ; with Adi Rome [and others].

Author	Harrison, Richard, 1946-
Published	Chichester, England ; Hoboken, NJ : Wiley, [2007]
Copyright Date	©2007
Physical Description	xxix, 803 pages : illustrations ; 24 cm
Additional Creators	Shackman, Mark

Availability

Browse Nearby on Shelf I Want It

Pattee Library and Paterno Library Stacks (1 item)

Call number	Material	Location
TK6570.M6H295 2007	Book	Paterno - 3rd Floor

Subject(s)

Cell phone systems—Computer programs
Operating systems (Computers)
C++ (Computer program language)

ISBN

9780470066416 (pbk.)
0470066415 (pbk.)

Bibliography Note

Includes bibliographical references and index.

Action Note

committed to retain 20170930 20421231 HathiTrust Shared Print Program https://www.hathitrust.org/shared_print_program

View MARC record | catkey: 3944836

ASK A LIBRARIAN

PennState

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

Legal Statements | PSU Hotlines

PENN STATE UNIVERSITY LIBRARIES
Libraries Home (814) 865-6368
Libraries Intranet (Staff Only)
Accessibility Help
Website Feedback
Policies and Guidelines
Staff Directory

CONNECT WITH PENN STATE UNIVERSITY LIBRARIES
Facebook
Twitter
Instagram

APPENDIX
HARRISON04

```
leader  01002cam a2200277 a 4500
001     991005670859708636
005     20071204142848.0
008     061107s2007    enka     b    001 0 eng d
015     ##$aGBA696235 $2bnb
016     7#$a013601779 $2Uk
020     ##$a9780470066416 (pbk.)
020     ##$a0470066415 (pbk.)
035     ##$a(CaOTR)b17679953
040     ##$aUKM $cUKM $dBAKER $dBTCTA $dBWKUK $dYDXCP $dCaOTR
050     #4$aTK6570.M6 $bH295 2007
100     1#$aHarrison, Richard, $d1946-
245     10$aSymbian OS C++ for mobile phones. $nVol. 3 / $cRichard Harrison, Mark Shackman ; with Adi Rome ... [et al.].
260     ##$aChichester, England ; $aHoboken, NJ : $bWiley, $cc2007.
300     ##$axxix, 803 p. : $bill. ; $c24 cm.
504     ##$aIncludes bibliographical references and index.
650     #0$aCell phone systems $xComputer programs.
650     #0$aOperating systems (Computers)
650     #0$aC++ (Computer program language)
700     1#$aShackman, Mark.
907     ##$a.b17679953 $b08-08-18 $c08-07-07 $9LOCAL
910     ##$atg
998     ##$amain0 $b12-04-07 $cm $db   $e- $feng $genk $h0
```



BOOK

Symbian OS C++ for mobile phones. Vol. 3

Harrison, Richard, 1946-; Shackman, Mark.
c2007

Available at Main Library 10th Floor (TK6570.M6 H295 2007)

TOP
SEND TO
DETAILS
GET IT
VIRTUAL BROWSE
LINKS

Send to

CITATION | PERMALINK | EXPORT RIS | ENDNOTE | EXPORT BIBTEX | PRINT

Details

Title	Symbian OS C++ for mobile phones. Vol. 3
Author/Creator	Harrison, Richard, 1946- ; Shackman, Mark. ;
Publication Date	c2007
Subject	Cell phone systems -- Computer programs ; Operating systems (Computers) ; C++ (Computer program language) ;
Identifier	ISBN : 9780470066416 (pbk.) ISBN : 0470066415 (pbk.)
Publisher	Chichester, England ; Hoboken, NJ : Wiley
Bibliography	Includes bibliographical references and index.
Format	xxix, 803 p. : ill. ; 24 cm.
Source	Library Catalog

Get It

Please sign in to check if there are any request options. Sign in

BACK TO LOCATIONS

LOCATION ITEMS

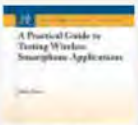
Main Library
Available , 10th Floor ; TK6570.M6 H295 2007
(1 copy, 1 available, 0 requests)

LOCATE

Item in place
Sign in to view the loan policy

Virtual Browse


dbook of arch on ile imedia ...


A practical guide to testing wireless smartphone ... c2010


Symbian OS C++ for mobile phones. Vol. 3 ... c2007


Interrogating user experiences with mobile phone ... c2009


Java ME or Symbian O c2009

Related reading

Recommended items that are related to the record


The C++ programming ...
Stroustrup, Bjarne.
ISBN 9780201700732
suggested by


The C++ programming ...
Stroustrup, Bjarne.
ISBN 9780201889543
suggested by


Effective C++ : 50 specific ways to ...
Meyers, Scott
ISBN 9780201563641
suggested by


C++ primer
Lippman, Stanley B
ISBN 9780321714114
suggested by


Beginning Java SE 6 Platform : From ...
Friesen, Jeff.
ISBN 9781430202462
suggested by

More

Links

Help with troubleshooting electronic resources access >
Report a problem with this record >
Use this form to report a problem or a broken link. Please retrieve the permalink for this record (above, in "Send to" section) and include it in your response.
Display Source Record >

Syndetics Unbound

Summary | About The Author | Look Inside
Book Profile

SUMMARY

Richard Harrison's existing books are the bestsellers in the Symbian Press Portfolio. His latest book, co-written with Mark Shackman is the successor to "Symbian OS C++ for Mobile Phones" Volumes One and Two. Written in the same style as the two previous volumes, this is set to be another gem in the series.
The existing material from the volumes will be combined, with explanations and (read more)

ABOUT THE AUTHOR



Richard Harrison

Mark Shackman is part of the Symbian Developer Network, which provides technical support to developers in the form of presentations, papers, books and tools. After the formation of Symbian, Mark joined the Connectivity Engineering group, with sole responsibility for authoring, producing, delivering and supporting the Connectivity SDK. He also wrote a chapter in Symbian's first book, *Professional* (read more)

More by Richard Harrison



MORE BY THIS AUTHOR

LOOK INSIDE

About the Authors ..... p. xi
Acknowledgements ..... p. xix
Symbian Press Acknowledgments ..... p. xxi

SEE FULL TABLE OF CONTENTS | COVER

BOOK PROFILE

Topics
Cellular telephones; Libraries; Ports; Smartphones; Software; Verbal communication; Wireless networks

Location
Japan; United Kingdom--UK; Europe; Sweden; Texas

Organization
Cambridge University; Microsoft Corp; Balliol College; Bluetooth Special Interest Group; Chase Manhattan Bank; Coventry University; JPMorgan Chase & Co; Middlesex University; Sussex University; Symbian Ltd

BROWSE PROFILE

Catalog enrichment powered by Syndetics Unbound

APPENDIX
HARRISON05

Source record page

```
leader  01432nam a2200373 a 4500
001     991014792069709636
005     20080326040945.0
008     061107t20072007enka     b    001 0 eng
015     ##$aGBA696235 $2bnb
016     7#$a013601779 $2Uk
020     ##$a9780470066416 $qpaperback
020     ##$a0470066415 $qpaperback
035     ##$a(NcRS)2045211
035     ##$a(OCoLC)76361852
040     ##$aUKM $beng $cUKM $dBAKER $dBTCTA $dBWKUK $dYDXCP $dNRC
049     ##$aNRCC $xsln
050     #4$aTK6570.M6 $bH295 2007
082     04$a621.38456028552768 $222
100     1#$aHarrison, Richard, $d1946- $0http://id.loc.gov/authorities/names/no2003067837
245     10$aSymbian OS C++ for mobile phones. $nVol. 3 / $cRichard Harrison, Mark Shackman ; with Adi Rome ... [and others].
264     #1$aChichester, England ; $aHoboken, NJ : $bWiley, $c[2007]
264     #4$c©2007
300     ##$axxix, 803 pages : $billustrations ; $c24 cm
336     ##$atext $btxt $2rdacontent
337     ##$aunmediated $bn $2rdamedia
338     ##$avolume $bnc $2rdacarrier
504     ##$aIncludes bibliographical references and index.
596     ##$a14
650     #0$aCell phone systems $xComputer programs.
650     #0$aOperating systems (Computers)
650     #0$aC++ (Computer program language)
700     1#$aShackman, Mark. $0http://id.loc.gov/authorities/names/n2007078347
916     ##$a2045211
999     ##$aTK6570 .M6 H295 2007 $wLC $c1 $d $e $f $iS02565167W $mBOOKBOT $j1 $kSTACKS $lSTACKS $n0 $p0 $q0 $u2008-01-14 $tBOOK $yBOOK
```



BOOK

Symbian OS C++ for mobile phones. Vol. 3

By Harrison, Richard, 1946-

Published: Chichester, England ; Hoboken, NJ : Wiley, 2007

Available at Hunt Library - bookBot Stacks (TK6570 .M6 H295 2007)

Locations

Hunt Library - bookBot

Available Stacks; TK6570 .M6 H295 2007 (1 copy, 1 available, 0 requests)

Available Loanable

Request this item

Sign in to view request options Sign in

Details

Title	**Symbian OS C++ for mobile phones**. Vol. 3
Creator	Harrison, Richard, 1946- Shackman, Mark.
Subject	Cell **phone** systems -- Computer programs Operating systems (Computers) C++ (Computer program language)
Publisher	Chichester, England ; Hoboken, NJ : Wiley
Creation Date	2007
Identifier	ISBN: 9780470066416 ISBN: 0470066415 OCLC: (OCoLC)76361852
Format	xxix, 803 pages : illustrations ; 24 cm
Other details	Includes bibliographical references and index.
Source	Library Catalog

Virtual Browse



Links

Display Source Record

Contact

D. H. Hill Jr. Library

2 Broughton Drive
Campus Box 7111
Raleigh, NC 27695-7111
(919) 515-3364

James B. Hunt Jr. Library

1070 Partners Way
Campus Box 7132
Raleigh, NC 27606-7132
(919) 515-7110

Libraries Administration

(919) 515-7188

NC State University Libraries

D. H. Hill Jr. Library
James B. Hunt Jr. Library
Design Library
Natural Resources Library
Veterinary Medicine Library
Accessibility at the Libraries
Accessibility at NC State University
Terms of Use
Jobs
Privacy Statement
Staff Confluence Login
Staff Drupal Login

Follow the Libraries

Facebook
Instagram
Threads
Bluesky
LinkedIn
Vimeo
YouTube
YouTube Archive
Flickr
Libraries' news

APPENDIX
HARRISON06

```
leader  00723nam a22002297a 4500
001     991002758199708046
005     20260520114748.0
008     061102s2007    enk    ||    |||| | eng|d
020     ##$a0470066415 (pbk.)
082     04$a005.133 $222
090     ##$aTK6570.M6 $kHARR
090     ##$aTK6570.M6
245     10$aSymbian OS C++ for mobile phones. $nvol. 3.
260     ##$aChichester :  $bWiley, $c2007.
300     ##$a416p. ; $c23.5 cm.
700     1#$aHarrison, Richard, $d1946-
700     1#$aShackman, Mark.
907     ##$a.b16085711 $b14-04-14 $c20-07-11
935     ##$a1590619
945     ##$g1 $i68071612 $j0 $lqm3   $o- $p£0.00 $q- $rs $s-   $t0 $u0 $v0 $w0 $x0 $y.i18465778 $z20-07-11
996     ##$aFull catalogue
998     ##$aqm $b20-07-11 $cm $da   $e- $feng $genk $h0
```



BOOK

Symbian OS C++ for mobile phones. vol. 3.

2007

Available at The McClay Library McClay Floor 3 (TK6570.M6 HARR)

TOP
SEND TO
GET IT
DETAILS
VIRTUAL BROWSE
LINKS

Send to

EXPORT BIBTEX | EXPORT TO ENDNOTE / RIS | EASYBIB | ENDNOTE | PRINT | CITATION | PERMALINK | EMAIL

Get It

Please sign in to check if there are any request options. Sign in

BACK TO LOCATIONS

LOCATION ITEMS

The McClay Library
Available , McClay Floor 3 ; TK6570.M6 HARR
(1 copy, 1 available, 0 requests)

Item in place
Loanable

Details

Title	Symbian OS C++ for mobile phones. vol. 3.
Creator	Harrison, Richard, 1946- Shackman, Mark.
Publisher	Chichester : Wiley
Publication Date	2007
Format	416p. ; 23.5 cm.
Source	Library Catalog
Identifier	ISBN : 0470066415 (pbk.)

Virtual Browse


...nces in ...ile radio ...ss ...orks ...


Symbian OS C[plus plus] for mobile phones ...
2004


Symbian OS C&& for mobile phones ...
2003


Symbian OS C++ for mobile phones. vol. 3. ...
2007


Cellular and PCS : the big picture ...
1997

Links

Display Source Record



APPENDIX
HARRISON07

```
leader  01202cam a2200337 a 4500
001     9962252923804341
005     20151125142231.0
008     061107s2007    enka     b    001 0 eng
035     ##$a(OCoLC)ocm76361852
035     ##$a(OCoLC)76361852
035     ##$a(StEdNL)6225292-nlsdb-Voyager
040     ##$aUKM $beng $cUKM $dBAKER $dBTCTA $dBWKUK $dYDXCP $dCQU $dOCLCA $dOCLCQ $dNLE
015     ##$aGBA696235 $2bnb
016     7#$a013601779 $2Uk
020     ##$a9780470066416 (pbk.)
020     ##$a0470066415 (pbk.)
050     #4$aTK6570.M6 $bH295 2007
082     04$a621.38456028552768 $222
049     ##$aNLES
100     1#$aHarrison, Richard, $d1946-
245     10$aSymbian OS C++ for mobile phones. $nVol. 3 / $cRichard Harrison, Mark Shackman ; with Adi Rome ... [et al.]
260     ##$aChichester : $bWiley, $cc2007.
300     ##$axxix, 803 pages : $billustrations ; $c24 cm
336     ##$atext $btxt $2rdacontent
337     ##$aunmediated $bn $2rdamedia
338     ##$avolume $bnc $2rdacarrier
500     ##$a"Application development for Symbian OS v9"--Cover.
504     ##$aIncludes bibliographical references and index.
650     #0$aCell phone systems $xComputer programs.
650     #0$aOperating systems (Computers)
650     #0$aC++ (Computer program language)
700     1#$aShackman, Mark.
```






Book

Symbian OS C++ for mobile phones. Vol. 3

Harrison, Richard, 1946-; Shackman, Mark.
Chichester; Wiley; c2007

Request

Top
Request
Details
Links

Export RIS | Export to Excel | Permalink | Email | Citation | Print

Request

Ask about copying

Log in to place requests. Log in

Locations:

Filters: location | volume | year | description | Clear filter

General Reading Room
Available , General Reading Room, Edinburgh (request to consult) ; PB4.207.256/1

Request from this location

Details

Title	Symbian OS C++ for mobile phones. Vol. 3
Author/Creator attribution	Richard Harrison, Mark Shackman ; with Adi Rome ... [et al.]
Name(s)	Harrison, Richard, 1946- Shackman, Mark,
Date	c2007
Publisher	Wiley
Place of publication	Chichester
General Note(s)	"Application development for Symbian OS v9"--Cover. Includes bibliographical references and index.
Physical description	xxix, 803 pages : illustrations ; 24 cm
Subject(s)	Cell phone systems -- Computer programs Operating systems (Computers) C++ (Computer program language)

Related Persons

Harrison, Richard, 1946-

Source Library of Congress Name Authority and Wikidata

Show less

Shackman, Mark

Show more

Language	English
ISBN	9780470066416 (pbk.) 0470066415 (pbk.)
Identifiers	OCLC : (OCoLC)ocm76361852 OCLC : (OCoLC)76361852 (StEdNL)6225292-nlsdb-Voyager
MMS ID	9962252923804341
Source	National Library of Scotland catalogue of published resources

Links

Display source record

APPENDIX
HARRISON08

```
leader  01216pam a2200325 a 4500
001     9945083343403606
005     20240620132004.0
008     061107s2007    enka     b    001 0 eng d
015     ##$aGBA696235 $2bnb
020     ##$a9780470066416 $cE36.99 $q(pbk.) :
020     ##$a0470066415 $cE36.99 $q(pbk.) :
035     ##$a(Uk)013601779
035     ##$a4508334
035     ##$a(UkCU)4508334-cambridgedb
040     ##$aStDuBDS $beng $cStDuBDS $dUk
042     ##$aukblsr
082     04$a621.38456 $222
100     1#$aHarrison, Richard, $d1946-
245     10$aSymbian OS C++ for mobile phones, volume 3 / $cRichard Harrison, Mark Shackman with Adi Rome ... [et al.] ; reviewed by Graeme Duncan ... [et al.] ; managing editor, Satu McNabb.
250     ##$a[New ed.]
260     ##$aChichester : $bWiley, $cc2007.
300     ##$axxix, 803 p. : $bill. ; $c24 cm.
500     ##$aSuccessor to Symbian OS C++ for mobile phones, vols. 1 and 2.
504     ##$aIncludes bibliographical references and index.
650     #0$aCell phone systems $xComputer programs.
650     #0$aOperating systems (Computers)
650     #0$aC++ (Computer program language)
700     1#$aShackman, Mark.
948     1#$a20071123 $blegd $cULLED-h $dc
948     2#$a20080618 $bsjs34 $cULCAT-h $db
```



BOOK

Symbian OS C++ for mobile phones, volume 3 / Richard Harrison, Mark Shackman with Adi Rome ... [et al.] ; reviewed by Graeme Duncan ... [et al.] ; managing editor, Satu McNabb.

Richard Harrison, 1946-; Mark Shackman
[New ed.]; Chichester : Wiley, c2007.

Available at University Library Order from Library Storage Facility (Not borrowable)

TOP
SEND TO
GET IT
DETAILS
LINKS

Send to

EXPORT TO EXCEL | READING LIST | EXPORT BIBTEX | RIS (ENDNOTE/ ZOTERO) | CITATION | PERMALINK | PRINT | E-MAIL

Get It

Please sign in to see request options and to place requests. Log in to iDiscover

BACK TO LOCATIONS

LOCATION ITEMS

University Library
Available , Order from Library Storage Facility (Not borrowable)
(1 copy, 1 available, 0 requests)

Item in place
Sign In to see options

Details

Title	Symbian OS C++ for mobile phones, volume 3 / Richard Harrison, Mark Shackman with Adi Rome ... [et al.] ; reviewed by Graeme Duncan ... [et al.] ; managing editor, Satu McNabb.
Author	Richard Harrison, 1946-
Other entry	Mark Shackman
Edition	[New ed.]
Publisher	Chichester : Wiley, c2007.
Identifier	ISBN : 9780470066416 (pbk.) : ISBN : 0470066415 (pbk.) :
Creation Date	2007
Format	xxix, 803 p. : ill. ; 24 cm.
Subjects	Cell phone systems -- Computer programs Operating systems (Computers) C++ (Computer program language)
Language	English
Notes	Successor to Symbian OS C++ for mobile phones, vols. 1 and 2 Includes bibliographical references and index.
Source	Cambridge Libraries Collections


Links
Report a problem
Table of Contents
This item in WorldCat®
Display Source Record

APPENDIX
HARRISON09

```
leader  01703cam a22003018a 4500
001     997041203503401
005     20170921002257.0
008     070724s2007    enka          000 0 eng d
015     ##$aGBA696235 $2bnb
020     ##$a9780470066416 (pbk.) : $c£36.99
020     ##$a0470066415 (pbk.) : $c£36.99
035     ##$a(Uk)013601779
035     ##$a(BNAtoc)B6802378
035     ##$a11115371
035     ##$a(NZ-HaUWL)704120-Waikatodb
035     ##$a(OCoLC)76361852
040     ##$aBNB
050     14$aTK6570 $b.M6H295 2007
082     04$a621.38456028552768 $222
245     00$aSymbian OS C++ for mobile phones. $nVol. 3 / $cRichard Harrison, Mark Shackman, editors.
260     ##$aChichester : $bWiley, $cc2007.
300     ##$axxix, 803 p. : $bill. ; $c24 cm.
505     00$g1. $tGetting started -- $g2. $tA system introduction to Symbian OS -- $g3. $tSymbian OS C++ -- $g4. $tObjects
650     #0$aCellular telephone systems $xComputer programs.
650     #0$aOperating systems (Computers)
650     #0$aC++ (Computer program language)
700     1#$aHarrison, Richard, $d1946-
700     1#$aShackman, Mark.
```



BOOK

Symbian OS C++ for mobile phones. Vol. 3

Chichester : Wiley; c2007

Available at Library General Collection Level 3 (TK6570 .M6H295 2007)

TOP
GET IT
SEND TO
DETAILS
LINKS

Get It

Please sign in to see request options and to place requests. Sign in

LOCATIONS:

Library
Available , General Collection Level 3 ; TK6570 .M6H295 2007 LOCATE

Send to

SHARE | QR | EXPORT BIBTEX | EXPORT RIS | CITATION | PERMALINK
PRINT | E-MAIL | EXPORT TO EXCEL

Details

Title	Symbian OS C++ for mobile phones. Vol. 3
Subjects	C++ (Computer program language) Cellular telephone systems -- Computer programs Operating systems (Computers)
Publisher	Chichester : Wiley
Publication Date	c2007
Contents	1. Getting started -- 2. A system introduction to Symbian OS -- 3. Symbian OS C++ -- 4. Objects - memory management, cleanup and error handling -- 5. Descriptors -- 6. Active objects -- 7. Files and the file system -- 8. Interprocess communication mechanisms -- 9. Platform security and publishing applications -- 10. Debugging and the emulator -- 11. The application framework -- 12. A simple graphical application -- 13. Resource files -- 14. Views and the view architecture -- 15. Controls -- 16. Dialogs -- 17. Graphics for display -- 18. Graphics for interaction -- 19. Plug-ins and extensibility -- 20. Communications and messaging services -- 21. Multimedia -- 22. Introduction to SQL RDBMS.
Format	xxix, 803 p. : ill. ; 24 cm.
Contributor	Harrison, Richard, 1946- Shackman, Mark.
Identifier	ISBN : 9780470066416 (pbk.) : ISBN : 0470066415 (pbk.) : OCLC : (OCoLC)76361852
MMS ID	997041203503401

Links

Display Source Record

Virtual Browse



net
ocol-
d
rgency
ces ...



Public safety networks from LTE to 5G ...

2020



Constant touch : a global history of the mobile phone ...

2004



Symbian OS C++ for mobile phones. Vol. 3 ...

c2007



Mobile interaction design ...

2006

APPENDIX
HARRISON10

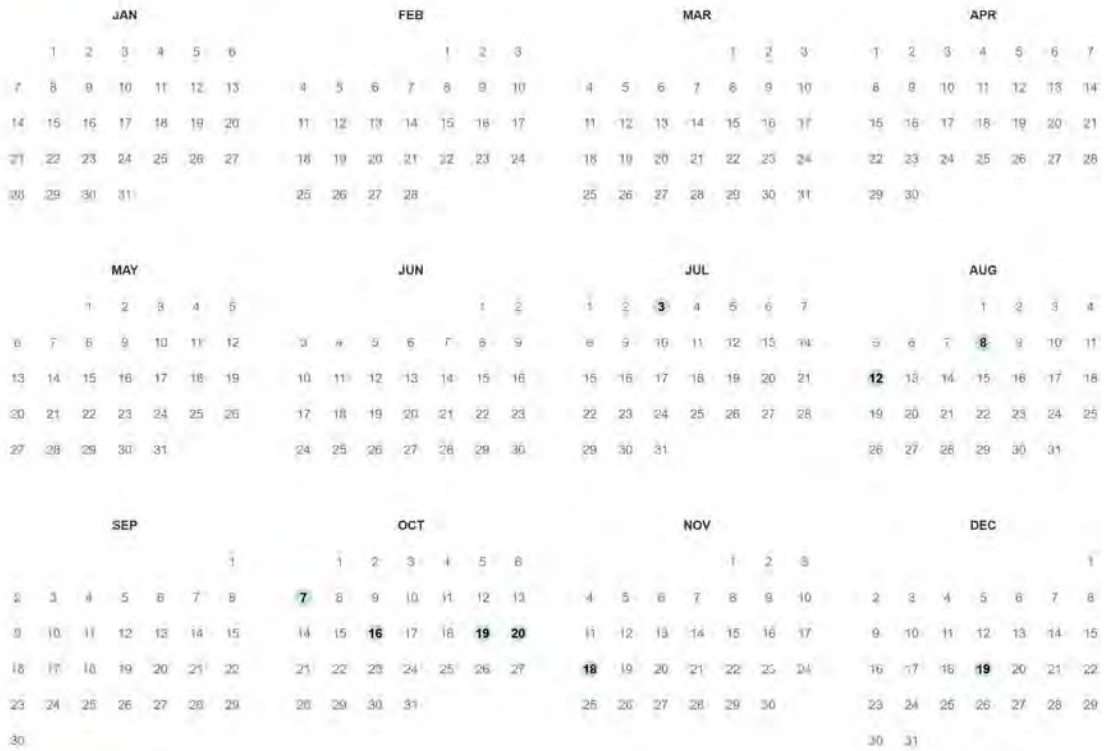
INTERNET ARCHIVE WEB TEXTS VIDEO AUDIO SOFTWARE All Search SIGN UP | LOG IN UPLOAD
ABOUT BLOG EVENTS PROJECTS HELP DONATE CONTACT JOBS VOLUNTEER

DONATE INTERNET ARCHIVE WaybackMachine Explore more than 1 trillion web pages saved over time
http://developer.symbian.com/main/learning/press/books/scmp_v

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved **16 times** between July 3, 2007 and June 22, 2008.





Note

This calendar view maps the number of times **http://developer.symbian.com/main/learning/press/books/scmp_v3/index.jsp** was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the FAQ.

FAQ | Contact Us | Terms of Service (Dec 31, 2014)

The Wayback Machine is an initiative of the Internet Archive, a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other projects include Open Library & archive-it.org.

Your use of the Wayback Machine is subject to the Internet Archive's Terms of Use

https://web.archive.org/web/20070703215621/http://developer.symbian.com/main/learning/press/books/scmp_v3/index.jsp July 30, 2026

INTERNET ARCHIVE WayBackMachine http://developer.symbian.com/main/learning/press/books/scmp_v3/index.jsp Go JUN JUL 03 2007 AUG 2006 2008
16 captures 3 Jul 2007 - 22 Jun 2008
About this capture

Home
Learning home
Symbian Press
Developer titles
Management titles
Translated titles
Essential Symbian OS booklets
Other publishers
A-Z index
Accredited Symbian Developer
FAQs
Training courses

Learning

Symbian OS C++ for Mobile Phones, Volume 3

Application Development for Symbian OS v9

Symbian OS C++ for Mobile Phones, Volume 3 is fully up-to-date for Symbian OS v9 and also combines the best of the two previous volumes, and includes sections about:

- An introduction to Symbian C++, with descriptions of the basic building blocks and usage patterns.
- Examples that are fully compatible with the latest Carbide.c++ Symbian IDEs guide the reader through increasingly complex aspects of Symbian C++.
- The core concepts, resources, APIs and programming idioms that you need to create, test and publish a non-trivial GUI (graphical user interface) application.
- An introduction to some of the most significant Symbian OS system services: extensibility and the use of plug-ins and practical descriptions of the communications, multimedia and database services.
- Advice on the available range of graphics APIs and helps you to ensure that your application code is as device-independent as possible.

Online: 3 users, 23 guests



Book facts

- Authors: Richard Harrison & Mark Shackman
- Published by Symbian Press/Wiley
- Publication date: June 2007
- 763 pages
- ISBN: 0470066415

Buy from amazon.com

Buy from WILEY

Related resources

Table of contents
Sample chapter
Source code

Bulk orders

To make a bulk order of this Symbian Press title:

Email Symbian Press sales

APPENDIX
HARRISON11

https://web.archive.org/web/20070000000000*/http://developer.symbian.com/main/learning/press/books/scmp_v3/toc.jsp July 30, 2026

INTERNET ARCHIVE WEB TEXTS VIDEO AUDIO SOFTWARE All Search SIGN UP | LOG IN UPLOAD

ABOUT BLOG EVENTS PROJECTS HELP DONATE CONTACT JOBS VOLUNTEER

DONATE

INTERNET ARCHIVE WayBackMachine

Explore more than 1 trillion web pages saved over time

http://developer.symbian.com/main/learning/press/books/scmp_v

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved **10 times** between October 23, 2007 and December 4, 2011.

003 2004 2005 2006 **2007** 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026

JAN

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

FEB

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28			

MAR

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

APR

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

MAY

		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

JUN

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JUL

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

AUG

			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

SEP

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

OCT

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	**23**	24	25	26	27
28	29	30	31			

NOV

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

DEC

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	**19**	20	21	22
23	24	25	26	27	28	29
30	31					

Note

This calendar view maps the number of times **http://developer.symbian.com/main/learning/press/books/scmp_v3/toc.jsp** was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the FAQ.

FAQ | Contact Us | Terms of Service (Dec 31, 2014)

The Wayback Machine is an initiative of the Internet Archive, a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other projects include Open Library & archive-it.org.

Your use of the Wayback Machine is subject to the Internet Archive's Terms of Use

https://web.archive.org/web/20071023075854/http://developer.symbian.com/main/learning/press/books/scmp_v3/toc.jsp July 30, 2026

INTERNET ARCHIVE WayBack Machine http://developer.symbian.com/main/learning/press/books/scmp_v3/toc.jsp Go SEP OCT DEC
10 captures 23 Oct 2007 - 4 Dec 2011 2006 23 2007 2009 About this capture

Home
Learning home
Symbian Press
Developer titles
Management titles
Translated titles
Booklets
Recommended reading
A-Z index
Accredited Symbian Developer
Chinese documents
Flash tutorials
FAQs
Training courses

Learning Resources

Symbian OS C++ for Mobile Phones, Volume 3

Application developement for Symbian OS v9

By Richard Harrison & Mark Shackman

Table of contents

1 Getting Started
1.1 Using the Emulator
1.2 Hello World – Text Version
Summary

2 A System Introduction to Symbian OS
2.1 Symbian OS Basics
2.2 Framework Basics
2.3 APIs Covered in this Book
Summary

3 Symbian OS C++
3.1 Fundamental Data Types
3.2 Naming Conventions
3.3 Functions
3.4 APIs
3.5 Templates
3.6 Casting
3.7 Classes
3.8 Design Patterns
Summary

4 Objects – Memory Management, Cleanup and Error Handling
4.1 Object Creation and Destruction
4.2 Class Categories in Symbian OS
4.3 Error Handling
4.4 The Cleanup Stack
4.5 Two-Phase Construction
Summary

5 Descriptors
5.1 Overview
5.2 Anatomy of Descriptors
5.3 Literals
5.4 Stack Descriptors
5.5 Pointer Descriptors
5.6 Heap Descriptors
5.7 Narrow, Wide and Neutral Descriptors
5.8 Descriptors and Binary Data
5.9 Using Descriptors with Methods
5.10 Some Descriptor Operations
5.11 Correct Use of Descriptors
5.12 Manipulating Descriptors
Summary

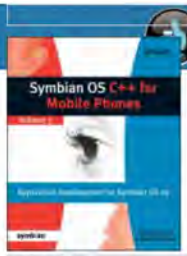
6 Active Objects
6.1 The Asynchronous Service
6.2 Multitasking and Pre-emption
6.3 A More In-depth Look at Active Objects
6.4 How It Works
6.5 Active Object Priorities
6.6 Active Object Cancellation
6.7 Starting and Stopping the Scheduler
6.8 Understanding a Stray Signal
6.9 Other Common Active Object Errors
6.10 Implementing State Machines
6.11 Long-Running Tasks and Active Objects
Summary

7 Files and the File System
7.1 File-Based Applications
7.2 Drives and File Types
7.3 File System Services
7.4 Streams
7.5 Stores
Summary

8 Interprocess Communication Mechanisms
8.1 Overview
8.2 Client–server IPC
8.3 Publish and Subscribe IPC
8.4 Message Queue IPC
8.5 Which IPC Mechanism Should You Use?
Summary

9 Platform Security and Publishing Applications
9.1 Releasing an Application
9.2 How Does Platform Security Work?
9.3 How Do I Support Platform Security?
9.4 Preparing an Application for Distribution
9.5 Overview of Symbian Signed
9.5 Installing a SIS File
9.6 List of Capabilities
Summary

10 Debugging and the Emulator
10.1 Using the Emulator
10.2 Emulator Debugging



Book facts

- Authors: Richard Harrison & Mark Shackman
- Published by Symbian Press/Wiley
- Publication date: June 2007
- 763 pages
- ISBN: 0470066415

Buy from amazon.com

Buy from WILEY

Related resources

Book overview
Sample chapter
Source code

Bulk orders

To make a bulk order of this Symbian Press title:

Email Symbian Press sales

10.3 Debugging on a Phone
10.4 Miscellaneous Tools
Summary

11 The Application Framework
11.1 Symbian OS Application Framework
11.2 S60 and UIQ Platform Application Frameworks
11.3 A Graphical Hello World
Summary

12 A Simple Graphical Application
12.1 Implementing the Game on S60
12.2 Differences for UIQ 3
12.3 Summary

13 Resource Files
13.1 Why a Symbian-Specific Resource Compiler?
13.2 Source File Syntax
13.3 Bitmaps and Icons
13.4 Updating the Resource Files
13.5 Application Registration Files
13.6 Localizable Strings
13.7 Multiple Resource Files
13.8 Compiling a Resource File
13.9 The Content of a Compiled Resource File
13.10 Reading Resource Files
Summary

14 Views and the View Architecture
14.1 The View Architecture
14.2 The MCoeView Interface
14.3 Introduction to the Example Application
14.4 Creating and Managing the Views
14.5 Implementing the MCoeView Interface
14.6 Command Menus
Summary

15 Controls
15.1 What Is a Control?
15.2 Control Types
15.3 Control Layout
15.4 Handling Key and Pointer Events
15.5 Observing a Control
15.6 Drawing a Control
15.7 Backed-up Windows
15.8 Backed-up-Behind Windows
15.9 Dimmed and Invisible Controls
Summary

16 Dialogs
16.1 What Is a Dialog?
16.2 Simple Dialogs
16.3 Complex Dialogs
16.4 Single-Page Dialogs
16.5 Multi-Page Dialogs
16.6 Dialog APIs
16.7 Stock Controls for Dialogs
16.8 Custom Controls in Dialogs
Summary

17 Graphics for Display
17.1 Drawing Basics
17.2 The CGraphicsContext API
17.3 Drawing and Redrawing
17.4 Drawing Controls
17.5 Sharing the Screen
17.6 Support for Drawing in CCoeControl
17.7 Special Effects
17.8 Window Server Features
17.9 Device- and Size-Independent Graphics
Summary

18 Graphics for Interaction
18.1 Key, Pointer and Command Basics
18.2 User Requirements for Interaction
18.3 Some Basic Abstractions
18.4 Processing Key Events
18.5 Processing Pointer Events
18.6 Window Server and Control Environment APIs
Summary

19 Plug-ins and Extensibility
19.1 System Services
19.2 What Is a Plug-in?
19.3 The ECOM Library
19.4 Plug-ins in Symbian OS
Summary

20 Communications and Messaging Services
20.1 Communications in Noughts and Crosses
20.2 Communication Between Controller and Transport
20.3 Serial Communications
20.4 Socket-based Communications
20.5 Messaging
20.6 Security
Summary

21 Multimedia
21.1 The Multimedia Framework
21.2 The Image Conversion Library
21.3 Camera API
21.4 Tuner API
Summary

22 Introduction to SQL RDBMS
22.1 Overview of RDBMS
22.2 SQL Basics
22.3 Symbian SQL Server Component Architecture
22.4 Symbian SQL Error Codes
Summary

Appendix
Developer Resources

Terms of use | Privacy policy | Media Center | Contact us | © 2007 Symbian

APPENDIX
HARRISON12

https://web.archive.org/web/20070000000000*/http://developer.symbian.com/main/learning/press/books/scmp_v3/c12.pdf July 30, 2026

INTERNET ARCHIVE WEB TEXTS VIDEO AUDIO SOFTWARE All Search SIGN UP | LOG IN UPLOAD

ABOUT BLOG EVENTS PROJECTS HELP DONATE CONTACT JOBS VOLUNTEER

DONATE

INTERNET ARCHIVE WaybackMachine

Explore more than 1 trillion web pages saved over time

http://developer.symbian.com/main/learning/press/books/scmp_v

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved **3 times** between October 23, 2007 and December 4, 2011.

003 2004 2005 2006 **2007** 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026

JAN

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

FEB

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28			

MAR

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

APR

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

MAY

		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

JUN

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JUL

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

AUG

			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

SEP

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

OCT

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	**23**	24	25	26	27
28	29	30	31			

NOV

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

DEC

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Note

This calendar view maps the number of times **http://developer.symbian.com/main/learning/press/books/scmp_v3/c12.pdf** was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the FAQ.

FAQ | Contact Us | Terms of Service (Dec 31, 2014)

The Wayback Machine is an initiative of the Internet Archive, a 501(c)(3) non-profit building a digital library of Internet sites and other cultural artifacts in digital form. Other projects include Open Library & archive-it.org.

Your use of the Wayback Machine is subject to the Internet Archive's Terms of Use

https://web.archive.org/web/20071023075829/http://developer.symbian.com/main/learning/press/books/scmp_v3/c12.pdf July 30, 2026
http://developer.symbian.com/main/learning/press/books/scmp_v3/c12.pdf Go SEP OCT JUL
3 captures 23 2007 2011
c12.dvi 1 / 36 100%

12

A Simple Graphical Application

After a series of chapters describing individual aspects of Symbian OS, it is time to pull a few threads together. In Chapter 11, we described a minimal GUI application that simply showed how the UI framework and the application UI fit together. Now we look at a complete, simple, but non-trivial example application that forms a basis for the examples used in several of the following chapters.

In this chapter, we cover some aspects of application design and introduce the idea of how to program a user interface that does not assume a particular screen size. This latter point is important if you want

12

A Simple Graphical Application

After a series of chapters describing individual aspects of Symbian OS, it is time to pull a few threads together. In Chapter 11, we described a minimal GUI application that simply showed how the UI framework and the application UI fit together. Now we look at a complete, simple, but non-trivial example application that forms a basis for the examples used in several of the following chapters.

In this chapter, we cover some aspects of application design and introduce the idea of how to program a user interface that does not assume a particular screen size. This latter point is important if you want to target your application at multiple user interfaces, as it helps speed up your porting efforts. In addition, we show how an application can use the principles described in Chapter 7 to save and restore its persistent data.

Before looking at the application's structure and behavior in more detail, we start with a general introduction to the Noughts and Crosses (otherwise known as Tic-Tac-Toe) application. Figure 12.1 shows the application running on a phone using the S60 user interface.

This application runs a game of noughts and crosses in which two players take turns to make their moves. The object of the game is to place three noughts (or crosses) in any row, column or diagonal before your opponent manages to do so.

We've implemented the game in a way that minimizes the differences needed to run on phones using the S60 and UIQ user interfaces. Depending on the nature of the phone it is running on, you can make a move either by using a combination of the cursor and pushbutton keys or by tapping on the screen.

FIFTH THIRD EX1062
PART 2 of 3
Page 108 of 379

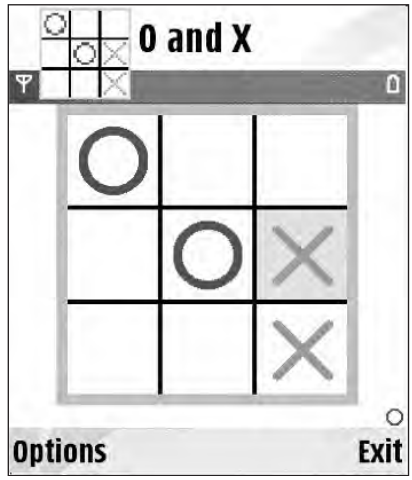


Figure 12.1 The Noughts and Crosses application

We've also kept the logic relating to the game itself very simple so that the bulk of the code illustrates issues of a general nature, relevant to most applications. In particular, we've designed the layout of the display in a way that can easily be adapted to show a rectangular grid with any number of tiles and to display on phones with a variety of screen formats. The grid layout is defined by a set of constants in `oandxdefs.h`:

```
static const TInt KTilesPerRow   = 3;
static const TInt KTilesPerCol   = 3;
static const TInt KNumberOfTiles = (KTilesPerRow*KTilesPerCol);
```

All the application's layout and drawing code is based on these values. By changing them, the layout can be adapted for games using differently sized and shaped boards.

The same header file also defines values that are specific to the Noughts and Crosses game, including the constant `KTilesPerSide` (which, unlike the earlier values, assumes that the board layout is square) and the states that can be associated with each tile:

```
static const TInt KTilesPerSide = 3;

enum TTileState
  {
  ETileBlank  = 0,
  ETileNought = 1,
  ETileCross  = 4
  };
```

The state values are chosen so that `ETileCross` is greater than the product of `ETileNought` with either `KTilesPerRow` or `KTilesPerCol`, which helps with the calculation of a winner.

This file also defines the height of the application's status window, used to show whose turn it is to play:

```
static const TInt KStatusWinHeight = 20; // in pixels
```

In Chapter 11 you saw that the simplest Symbian OS graphical application needs instances of four classes: the application, the document, the application UI and a view. For this game, we've added two more classes: a controller and a model, or engine. Since the game control logic is so simple, we could have implemented it entirely within the application UI class, rather than defining separate controller and engine classes. However, we've chosen to add these classes from the start, in preparation for the more complex form of the game that is developed in Chapter 20.

The overall structure of the game is shown in Figure 12.2.

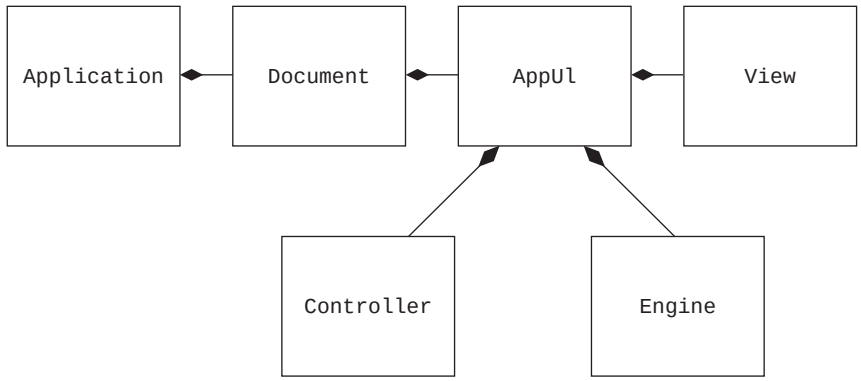


Figure 12.2 Structure of the Noughts and Crosses application

We cover the S60 version of the game first and then look at the changes that are needed to create the UIQ version.

12.1 Implementing the Game on S60

The Application Class

The S60 class definition for the Noughts and Crosses application class is listed below. As you can see, it is very similar to the corresponding class in Chapter 11's basic application: it implements `AppDllUid()`, to report the application's UID to the application framework, and `CreateDocumentL()`.

```
class COandXApplication : public CAknApplication
  {
public:
  // From CApaApplication
  TUid AppDllUid() const;

protected:
  // From CEikApplication
  CApaDocument* CreateDocumentL();
  };
```

This is fairly typical for applications. It is rare for an application class to override any of the other application class member functions.

The Document Class

The class definition for the Noughts and Crosses document class, listed below, provides the implementation of `CreateAppUiL()`, which is always required. In addition, it implements persistent storage of the game's state in the `StoreL()` and `RestoreL()` functions, using the principles described in Chapter 7.

```
class COandXDocument : public CAknDocument
  {
public:
  static COandXDocument* NewL(CEikApplication& aApp);
  virtual ~COandXDocument();

  // From CEikDocument
  CEikAppUi* CreateAppUiL();
  void StoreL(CStreamStore& aStore, CStreamDictionary& aStreamDic) const;
  void RestoreL(const CStreamStore& aStore,
      const CStreamDictionary& aStreamDic);
  CFileStore* OpenFileL(TBool aDoOpen, const TDesC& aFilename, RFs& aFs);
private:
  COandXDocument(CEikApplication& aApp);
private:
  COandXAppUi* iAppUi;
  };
```

The implementations of `StoreL()` and `RestoreL()` delegate to the application UI class the actual task of storing and restoring the application's data, but they retain the responsibility for associating the application's UID with the ID of the stream in which the data is stored.

```
void COandXDocument::StoreL(CStreamStore& aStore,
            CStreamDictionary& aStreamDic) const
  {
  TStreamId id = iAppUi->StoreL(aStore);
```

```
  aStreamDic.AssignL(KUidOandXApp, id);
  }

void COandXDocument::RestoreL(const CStreamStore& aStore,
                    const CStreamDictionary& aStreamDic)
  {
  TStreamId id = aStreamDic.At(KUidOandXApp);
  iAppUi->RestoreL(aStore, id);
  }
```

The application framework provides support for an application to save its document data in an application-specific direct file store and, in UIQ, it is sufficient just to implement the two functions listed above.

In contrast, S60 applications are assumed not to be document-based and the default behavior is that the application framework never calls these two functions. In order to restore the support, you need to override the `OpenFileL()` function to call `CEikDocument`'s implementation of `OpenFileL()`:

```
CFileStore* COandXDocument::OpenFileL(TBool aDoOpen,
                  const TDesC& aFilename, RFs& aFs)
  {
  return CEikDocument::OpenFileL(aDoOpen, aFilename, aFs);
  }
```

The Application UI Class

The application UI class normally forms the core of an application, and the `COandXAppUi` class is no exception. In addition to its standard role of handling much of the interaction with the user, `COandXAppUi` owns the application's view, controller and engine instances and supplies the central logic for saving and restoring the application's persistent data.

```
class COandXAppUi : public CAknAppUi
  {
public:
  COandXAppUi();
  virtual ~COandXAppUi();

  // New functions
  void ReportWhoseTurn();
  void ReportWinnerL(TInt aWinner);

  // From CEikAppUi, for persistent data
  TStreamId StoreL(CStreamStore& aStore) const;
  void RestoreL(const CStreamStore& aStore, TStreamId aStreamId);
  void ExternalizeL(RWriteStream& aStream) const;
  void InternalizeL(RReadStream& aStream);

private:
  // From MEikMenuObserver
```

```
  void DynInitMenuPaneL(TInt aResourceId, CEikMenuPane* aMenuPane);
  // From CEikAppUi
  void HandleCommandL(TInt aCommand);
  void ConstructL();

public:
  // AppUi owns the engine, controller and application view.
  COandXEngine* iEngine;
  COandXController* iController;
private:
  COandXAppView* iAppView;
  // Set if the application view was added to the control stack.
  TBool iStacked;
  };
```

As you can see from the above class definition, part of the user interaction consists of reporting significant events such as whose turn it is and, at the end of a game, who (if anyone) has won. In the case of the S60 version, the application UI class also handles all aspects of using menu commands, including dynamically setting the menu content according to the current state of the game. We have more to say about using menus later in this chapter.

The second-phase constructor and the destructor make it clear that the application UI class owns the engine, the controller and the view, which are all created in a leave-safe way:

```
void COandXAppUi::ConstructL()
  {
  BaseConstructL(EAknEnableSkin);
  iEngine = COandXEngine::NewL();
  iController = COandXController::NewL();
  iAppView = COandXAppView::NewL(ClientRect());
  AddToStackL(iAppView); // Enable keypresses to the view
  iStacked = ETrue;
  ReportWhoseTurn();
  }

COandXAppUi::~COandXAppUi()
  {
  if (iStacked)
    {
    RemoveFromStack(iAppView);
    }
  delete iAppView;
  delete iController;
  delete iEngine;
  }
```

It is worth noting that, it is the responsibility of the application UI class to ensure that the view is added to the control stack, via a call to `AddToStackL()`, to ensure that it has the opportunity to receive and process keypresses. Since this call can fail, it is necessary to set a flag if

it succeeds, so that the flag can be tested (in this case, since there is only one view, in the destructor) before the view is removed from the control stack.

This application uses two different ways to report events to the user. Reporting whose turn it is to play, which happens between every move in the game, is delegated to the view:

```
void COandXAppUi::ReportWhoseTurn()
  {
  iAppView->ShowTurn();
  }
```

If either player wins, which obviously happens no more than once per game, the application UI class reports the winner by means of a UI-specific dialog that displays the appropriate text, read from the application's resource file. The S60 version uses an information note:

```
void COandXAppUi::ReportWinnerL(TInt aWinner)
  {
  TBuf<MaxInfoNoteTextLen> text;
  iEikonEnv->ReadResource(text, aWinner==ETileCross
               ? R_OANDX_X_WINS : R_OANDX_O_WINS);
  CAknInformationNote* infoNote = new (ELeave) CAknInformationNote;
  infoNote->ExecuteLD(text);
  }
```

The application's persistent data is saved and restored via the `StoreL()` and `RestoreL()` functions of the application UI class which, you may recall, are called from the application's document class.

```
TStreamId COandXAppUi::StoreL(CStreamStore& aStore) const
  {
  RStoreWriteStream stream;
  TStreamId id = stream.CreateLC(aStore);
  stream << *this; // alternatively, use ExternalizeL(stream)
  stream.CommitL();
  CleanupStack::PopAndDestroy();
  return id;
  }

void COandXAppUi::RestoreL(const CStreamStore& aStore,
                                  TStreamId aStreamId)
  {
  RStoreReadStream stream;
  stream.OpenLC(aStore, aStreamId);
  stream >> *this; // alternatively use InternalizeL(stream)
  CleanupStack::PopAndDestroy();
  }
```

`Store()` and `Restore()`, respectively, call the `ExternalizeL()` and `InternalizeL()` functions of the application UI class via the

<< and >> operators. As you can see from the following listing, the application UI class itself has no persistent data. The functions deal directly with the view's data, which is simply the ID of the control that currently has focus, and leave the engine and the controller to take care of their own data.

```
void COandXAppUi::ExternalizeL(RWriteStream& aStream) const
  {
  iEngine->ExternalizeL(aStream);
  iController->ExternalizeL(aStream);
  aStream.WriteInt8L(iAppView->IdOfFocusControl());
  }

void COandXAppUi::InternalizeL(RReadStream& aStream)
  {
  iEngine->InternalizeL(aStream);
  iController->InternalizeL(aStream);
  ReportWhoseTurn();
  iAppView->MoveFocusTo(aStream.ReadInt8L());
  }
```

The S60 application UI class is also responsible for handling menu commands, via the `DynInitMenuPaneL()` and `HandleCommandL()` functions. We discuss them – and the differences between the ways they are handled in S60 and UIQ – later in this chapter.

Finally, it is worth pointing out that we provide access to the controller and engine via global static functions, declared (in `oandxappui.h`) as:

```
GLREF_C COandXAppUi* OandXAppUi();
GLREF_C COandXController& Controller();
GLREF_C COandXEngine& Engine();
```

The implementations of these functions are:

```
GLDEF_C COandXAppUi* OandXAppUi()
  {
  return static_cast<COandXAppUi*>(CEikonEnv::Static()->AppUi());
  }

GLDEF_C COandXController& Controller()
  {
  return *OandXAppUi()->iController;
  }

GLDEF_C COandXEngine& Engine()
  {
  return *OandXAppUi()->iEngine;
  }
```

We have chosen to use global functions in this example because they accurately represent the intention that these functions should be

accessible from anywhere in the program. Furthermore, they do not need verbose source text to call them, which is useful in example code. We would not use them in production code, since there is no control of when and from where they are called. Additionally, calling a global function involves accessing thread-local storage, which is slow compared with other function calls. We could improve the efficiency by implementing them as static members of a class, but this relies on the availability of writable static data and so would not work on all phones. The best solution would be to implement them as normal member functions of, for example, the application UI class and pass a (non-owning) reference to the constructor of each class that needs to access them. We have not done that in this example because it would add a significant amount of incidental complexity to each of the constructors concerned.

The Controller Class

The controller class, whose definition is listed below, is responsible for managing the state of the game and its logic, including updating the data in the engine class.

```
class COandXController : public CBase
  {
public:
  static COandXController* NewL();
  virtual ~COandXController();
enum TState
  {
  ENewGame, EPlaying, EFinished
  };
  // state
  inline TBool IsNewGame() const;
  // game control
  void Reset();
  // stream persistence
  void ExternalizeL(RWriteStream& aStream) const;
  void InternalizeL(RReadStream& aStream);

  TBool HitSquareL(TInt aIndex);
  TBool IsCrossTurn() const;
  void SwitchTurn();
private:
  void ConstructL();
private:  // private persistent state
  TState iState;
  TBool iCrossTurn;
  };
```

In this stand-alone, non-communicating, version, the game state is simply one of the three values `ENewGame`, `EPlaying` or `EFinished`, together with a flag that indicates which player is next to play. The only externally accessible state-query functions needed are `IsCrossTurn()`,

which simply returns the value of `iCrossTurn`, and `IsNewGame()`, coded as:

```
inline TBool COandXController::IsNewGame() const
  { return iState==ENewGame; }
```

The central logic of the controller is contained in the `HitSquareL()` function:

```
TBool COandXController::HitSquareL(TInt aIndex)
  {
  if (iState == EFinished)
    {
    return EFalse;
    }
  if (iState == ENewGame)
    {
    iState = EPlaying;
    }
  if (Engine().TryMakeMove(aIndex, IsCrossTurn()))
    {
    SwitchTurn();
    TInt winner = Engine().GameWonBy();
    if (winner)
      {
      iState = EFinished;
      OandXAppUi()->ReportWinnerL(winner);
      }
    return ETrue;
    }
  return EFalse;
  }
```

This function is called from the application's board view whenever a player attempts to make a move. It disallows the move if the game state is `EFinished`; if this is the first move in the game (i.e. the game state is `ENewGame`), it sets the state to `EPlaying`. If the game is in play, it calls the `TryMakeMove()` function of the engine class to check if the move is valid; if so, it records the move and switches whose turn it is. A call to the `GameWonBy()` function of the engine class checks if the last move resulted in a win by either player and, if it did, the game state is set to `EFinished` and a call to the `ReportWinner()` function of the application UI class displays the result.

In its current form, `HitSquareL()` does not check for, or report, a drawn game. If there is no winner, the game continues until all tiles contain a nought or a cross (or a player uses a menu item to start a new game). The functionality for a drawn game is added in the communicating version of the game, described in Chapter 20.

The controller is reset for a new game by calling `Reset()`, which cancels the current game, clears the board (by calling the `Reset()` function of the engine class) and sets Noughts as the current player:

FIFTH THIRD EX1062
PART 2 of 3
Page 117 of 379

```
void COandXController::Reset()
  {
  Engine().Reset();
  iState = ENewGame;
  if (IsCrossTurn())
    {
    SwitchTurn();
    }
  }
```

The `SwitchTurn()` function simply negates the flag held in `iCrossTurn` and calls the `ReportWhoseTurn()` function of the application UI class:

```
void COandXController::SwitchTurn()
  {
  iCrossTurn = !iCrossTurn;
  OandXAppUi()->ReportWhoseTurn();
  }
```

Persistence of the state of the controller class is handled by the `ExternalizeL()` and `InternalizeL()` functions, which are called from the application UI class. Their actions are, respectively, to write and read the controller's member data to or from the passed stream:

```
void COandXController::ExternalizeL(RWriteStream& aStream) const
  {
  aStream.WriteUint8L(iState);
  aStream.WriteInt8L(iCrossTurn);
  }

void COandXController::InternalizeL(RReadStream& aStream)
  {
  iState = static_cast<TState>(aStream.ReadUint8L());
  iCrossTurn = static_cast<TBool>(aStream.ReadInt8L());
  }
```

The controller is created via a call to its static `NewL()` function, which is coded as:

```
COandXController* COandXController::NewL()
  {
  COandXController* self=new(ELeave) COandXController;
  CleanupStack::PushL(self);
  self->ConstructL();
  CleanupStack::Pop();
  return self;
  }
```

The second-phase constructor is simply:

```
void COandXController::ConstructL()
  {
  Reset();
  }
```

We have coded it in this way to preserve the standard form of class construction, as described in Chapter 4. However, since – in this case – `ConstructL()` does not leave, it is worth pointing out that we could have simplified matters by not defining a `ConstructL()` function and doing everything in the `NewL()` function.

The Engine Class

The engine class, whose class definition is listed below, represents the game data and contains the functions that operate on that data. Since Noughts and Crosses is a simple game, the engine class is correspondingly straightforward. The class itself has no knowledge of the overall state of the game, nor of whose turn it is to make a move. As you saw in the previous section, both of these are the responsibility of the controller.

```
class COandXEngine : public CBase
  {
public:
  static COandXEngine* NewL();
  virtual ~COandXEngine();

  void Reset();
  TInt TileStatus(TInt aIndex) const;
  TBool TryMakeMove(TInt aIndex, TBool aCrossTurn);
  TTileState GameWonBy() const;
  // persistence
  void ExternalizeL(RWriteStream& aStream) const;
  void InternalizeL(RReadStream& aStream);
private:
  COandXEngine();
  TInt TileState(TInt aX, TInt aY) const;
private:
  TFixedArray<TTileState, KNumberOfTiles> iTileStates;
  };
```

The current state of the board is represented by a simple array of tile states, one for each component tile. Each tile state contains one of the three values `ETileBlank`, `ETileNought` or `ETileCross`.

The key functions are `TryMakeMove()` and `GameWonBy()`, both of which are called from the controller. `TryMakeMove()` only allows a move to be made if the corresponding tile is initially blank. If the move is allowed, the function sets the tile-state variable corresponding to the

tile to either `ETileCross` or `ETileNought`, depending on the value passed in `aCrossTurn`.

```
TBool COandXEngine::TryMakeMove(TInt aIndex, TBool aCrossTurn)
  {
  if (iTileStates[aIndex] == ETileBlank)
    {
    iTileStates[aIndex] = aCrossTurn ? ETileCross : ETileNought;
    return ETrue;
    }
  return EFalse;
  }
```

`GameWonBy()` checks if there is a winner and, if so, whether the winner is the Nought player or the Cross player. It does so by determining if there is a horizontal, vertical or diagonal line of three noughts or three crosses. If there is such a line, it returns either `ETileNought` or `ETileCross`; otherwise it returns zero, to indicate that, as yet, there is no winner.

```
TTileState COandXEngine::GameWonBy() const
  {
  const TInt KNoughtWinSum = KTilesPerSide * ETileNought;
  const TInt KCrossWinSum = KTilesPerSide * ETileCross;

  // is there a row or column of matching tiles?
  for (TInt i = 0; i < KTilesPerSide; ++i)
    {
    TInt rowSum = 0;
    TInt colSum = 0;
    for (TInt j = 0; j < KTilesPerSide; ++j)
      {
      rowSum += TileState(j, i);
      colSum += TileState(i, j);
      }
    if (rowSum == KNoughtWinSum || colSum == KNoughtWinSum)
      return ETileNought;
    if (rowSum == KCrossWinSum || colSum == KCrossWinSum)
      return ETileCross;
    }
  // is there a diagonal of matching tiles?
  TInt blTrSum = 0;        // bottom left to top right
  TInt tlBrSum = 0;        // top left to bottom right
  for (TInt i = 0; i < KTilesPerSide; ++i)
    {
    tlBrSum += TileState(i, i);
    blTrSum += TileState(i, KTilesPerSide - 1 - i);
    }
  if (blTrSum == KNoughtWinSum || tlBrSum == KNoughtWinSum)
    return ETileNought;
  if (blTrSum == KCrossWinSum || tlBrSum == KCrossWinSum)
    return ETileCross;
  return ETileBlank;  // No winner
  }
```

As with the controller class, the `ExternalizeL()` and `InternalizeL()` functions of the engine class, called from the application UI class, simply write and read the state – in this case, the values of each of the board's tiles – to and from the appropriate stream:

```
void COandXEngine::ExternalizeL(RWriteStream& aStream) const
  {
  for (TInt i = 0; i<KNumberOfTiles; i++)
    {
    aStream.WriteInt8L(iTileStates[i]);
    }
  }

void COandXEngine::InternalizeL(RReadStream& aStream)
  {
  for (TInt i = 0; i<KNumberOfTiles; i++)
    {
    iTileStates[i] = static_cast<TTileState>(aStream.ReadInt8L());
    }
  }
```

Like the controller class, the engine class is constructed by means of a call to its static `NewL()` function and, again, there are no potentially leaving calls to be made from a second-phase constructor. For the engine class, we've chosen to illustrate an alternative way of implementing construction in such a case, where the non-leaving calls are made from the constructor rather than from the `NewL()` function:

```
COandXEngine* COandXEngine::NewL()
  {
  return new(ELeave) COandXEngine;
  }

COandXEngine::COandXEngine()
  {
  Reset();
  }
```

The View Class

Views and the view architecture are discussed in Chapter 14 and the detailed behavior of controls is described in Chapters 15, 17 and 18. In this chapter, therefore, we only briefly describe the general aspects of controls and views, concentrating on those features that are specific to the Noughts and Crosses application.

An application's view is an instance of a window-owning control (as defined in Chapter 15) and derives, directly or indirectly, from the `CCoeControl` class. For the Noughts and Crosses application, the view is a compound control, with the view's area being tiled with a number of subsidiary controls.

FIFTH THIRD EX1062
PART 2 of 3
Page 121 of 379

We've chosen to implement the S60 view class by deriving directly from `CCoeControl`. S60 applications can derive from more specific classes, such as `CAknView`, but we decided not to for this application in order to reduce the differences between the S60 and UIQ versions, and because many of the S60-specific views are more suited to text displays. The view also inherits from an `MViewCmdHandler` interface class, which we describe later.

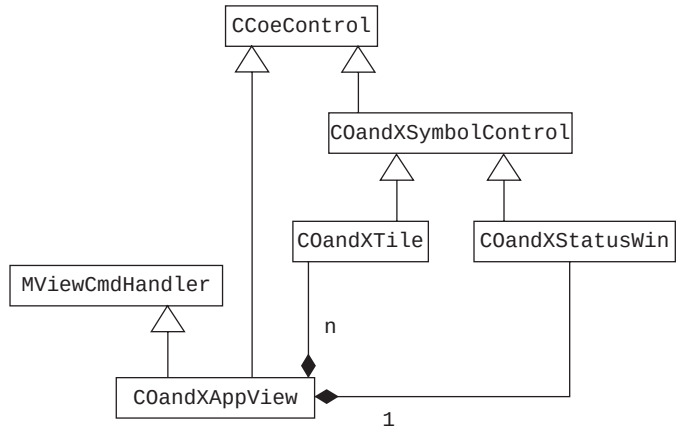


Figure 12.3 The View class of the Noughts and Crosses application

Figure 12.3 shows that all the components of the view are themselves also controls, in the sense that they all derive from `CCoeControl`. The view owns a single status-display control and a series of tiles.

Both the status display control and the tiles need to display either a nought or a cross, so we've chosen to implement them as subclasses of a generic symbol-drawing control, which knows how to draw those symbols. The two subclasses simply add the specification of the size and location, within their own area, of the symbol.

As you can see from the class definition of the symbol-drawing base class, its only member function is to draw a nought or a cross symbol:

```
class COandXSymbolControl : public CCoeControl
  {
protected:
  void DrawSymbol(CWindowGc& aGc, const TRect& aRect,
                              TBool aDrawCross) const;
  };
```

The implementation of this function is:

```
void COandXSymbolControl::DrawSymbol(CWindowGc& aGc,
        const TRect& aRect, TBool aDrawCross) const
```

```
  {
  TRect drawRect(aRect);

  // Shrink by about 15%
  drawRect.Shrink(aRect.Width()/6, aRect.Height()/6);

  // Pen size set to just over 10% of the shape's size
  TSize penSize(aRect.Width()/9, aRect.Height()/9);
  aGc.SetPenSize(penSize);
  aGc.SetPenStyle(CGraphicsContext::ESolidPen);

  if (aDrawCross)
    {
    aGc.SetPenColor(KRgbGreen);

    // Cosmetic reduction of cross size by half the line width
    drawRect.Shrink(penSize.iWidth/2, penSize.iHeight/2);

    aGc.DrawLine(drawRect.iTl, drawRect.iBr);
    TInt temp;
    temp = drawRect.iTl.iX;
    drawRect.iTl.iX = drawRect.iBr.iX;
    drawRect.iBr.iX = temp;
    aGc.DrawLine(drawRect.iTl, drawRect.iBr);
    }
  else // draw a circle
    {
    aGc.SetPenColor(KRgbRed);
    aGc.SetBrushStyle(CGraphicsContext::ESolidBrush);
    aGc.DrawEllipse(drawRect);
    }
  };
```

You see this code again in Chapter 15, where the more detailed aspects of drawing this control are discussed; here, it is sufficient to note that it is written to be as independent as possible of the size or position of the symbol to be drawn. The fact that the same code successfully draws both the large symbols in the tiles and the much smaller symbols in the status area shows that this approach is both feasible and sensible.

The symbol is centered in the rectangle (which is always a square) passed to the function in `aRect`, but shrunk to leave a small border, and the pen width is set to match the size of the symbol. Note that the cross symbol is shrunk by a further small amount since, if the circles and crosses were both drawn to exactly the same size, the cross would look larger to the eye. This kind of attention to detail can make a large difference to the appearance of any application.

The tile and status window control classes both derive from `COandX-SymbolControl`:

```
class COandXTile : public COandXSymbolControl
  {
public:
  COandXTile();
```

```
  ~COandXTile();
  void ConstructL(RWindow& aWindow);
  // New function
  void SetOwnerAndObserver(COandXAppView* aControl);
  // From CCoeControl
  TKeyResponse OfferKeyEventL(const TKeyEvent& aKeyEvent,
                                    TEventCode aType);
  TCoeInputCapabilities InputCapabilities() const;
protected:
  void FocusChanged(TDrawNow aDrawNow);
  void HandlePointerEventL(const TPointerEvent& aPointerEvent);
private:
  void Draw(const TRect& aRect) const;
  // New function
  void TryHitL();
private:
  MViewCmdHandler* iCmdHandler;
  };

class COandXStatusWin : public COandXSymbolControl
  {
public:
  static COandXStatusWin* NewL(RWindow& aWindow);
  ~COandXStatusWin();
private:
  COandXStatusWin();
  void ConstructL(RWindow& aWindow);
  void Draw(const TRect& aRect) const;
  }
```

Note that, even though S60 devices do not normally use touch screens, we include code that handles input from keys – via `OfferKeyEventL()` – and any pointer device – via `HandlePointerEventL()`. It is quite safe to do this: if a particular device does not support one or the other input device, the relevant code simply is not called. Including both options makes it easier to port the code to different devices.

The status window's `Draw()` function simply calculates the square in which to draw the symbol and then calls `DrawSymbol()`, obtaining the flag indicating which symbol to draw by calling the `IsCrossTurn()` function of the controller class:

```
void COandXStatusWin::Draw(const TRect& /*aRect*/) const
  {
  CWindowGc& gc = SystemGc();
  TRect boxRect = Rect();
  gc.Clear(boxRect);
  TInt boxHeight = boxRect.iBr.iY - boxRect.iTl.iY;
  boxRect.iTl.iX = boxRect.iBr.iX - boxHeight;
  DrawSymbol(gc, boxRect, Controller().IsCrossTurn());
  }
```

FIFTH THIRD EX1062
PART 2 of 3
Page 124 of 379

The `Draw()` function for each tile works in a similar way:

```
void COandXTile::Draw(const TRect& /*aRect*/) const
  {
  TTileState tileType;
  tileType = iCmdHandler->TileStatus(this);

  CWindowGc& gc = SystemGc();
  TRect rect = Rect();

  if (IsFocused())
    {
    gc.SetBrushColor(KRgbYellow);
    }
  gc.Clear(rect);
  if (tileType!=ETileBlank)
    {
    DrawSymbol(gc, rect, tileType==ETileCross);
    }
  }
```

In this case, the symbol to be drawn is determined by a call to the `TileStatus()` function of the view class, by a mechanism that is described later. The tile which currently has focus is drawn with a yellow background; the others are drawn with the default brush color, which is white.

During the game, a player places a nought or a cross in a particular tile either by tapping on it or by using cursor keys to move the highlight to that tile and pressing the phone's Selection or Action key. A pointer event is handled by the tile's `HandlePointerEventL()` function:

```
void COandXTile::HandlePointerEventL (const TPointerEvent& aPointerEvent)
  {
  if (aPointerEvent.iType == TPointerEvent::EButton1Down)
    {
    TryHitL();
    }
  }
```

A key event is handled by `OfferKeyEventL()`:

```
TKeyResponse COandXTile::OfferKeyEventL(const TKeyEvent& aKeyEvent,
                                                    TEventCode aType)
  {
  TKeyResponse keyResponse = EKeyWasNotConsumed;
  if (aType!=EEventKey)
    {
    return keyResponse;
    }
  switch (aKeyEvent.iCode)
    {
    case EKeyOK:
```

```
      TryHitL();
      keyResponse = EKeyWasConsumed;
      break;
    default:
      keyResponse = EKeyWasNotConsumed;
      break;
    }
  return keyResponse;
  }
```

In both cases, this results in a call to the tile's `TryHitL()` function, which is implemented as:

```
void COandXTile::TryHitL()
  {
  if (iCmdHandler->TryHitSquareL(this))
    {
    DrawDeferred();
    }
  }
```

`TryHitL()` calls the `TryHitSquareL()` function of the view class, using the same mechanism that is also used to call `TileStatus()`.

The `COandXAppView` class definition has 19 member functions, not counting those that it inherits from its base classes, but only the two mentioned above are called from a tile. As an encapsulation aid, we have chosen to declare these two functions, as pure virtual functions, in an `MViewCmdHandler` interface class:

```
class MViewCmdHandler
  {
public:
  virtual TBool TryHitSquareL(const COandXTile* aControl) = 0;
  virtual TTileState TileStatus(const COandXTile* aControl) const = 0;
  };
```

The view class inherits from this interface and provides the concrete implementations, as indicated by the bold portions of the `COandXAppView` class definition:

```
class COandXAppView : public CCoeControl, public MCoeControlObserver,
                                                public MViewCmdHandler
  {
public:
  static COandXAppView* NewL(const TRect& aRect);
  virtual ~COandXAppView();

  // From CCoeControl
  TKeyResponse OfferKeyEventL(const TKeyEvent& aKeyEvent,
                                       TEventCode aType);
```

```
  // new functions
  void MoveFocusTo(const TInt aIndex);
  TInt IdOfFocusControl();
  void ShowTurn();
  void ResetView();

private:
  COandXAppView();
  void ConstructL(const TRect& aRect);

  void SwitchFocus(TInt aFromIndex, CCoeControl* aToControl);
  void DrawComps(TRect& aRect) const;
  COandXTile * CreateTileL();

  // From CCoeControl
  void Draw(const TRect& aRect) const;
  void SizeChanged();
  TInt CountComponentControls() const;
  CCoeControl* ComponentControl(TInt aIndex) const;

  // From MCoeControlObserver
  void HandleControlEventL(CCoeControl* aControl, TCoeEvent aEventType);

  // From MViewCmdHandler
  TBool TryHitSquareL(const COandXTile* aControl);
  TTileState TileStatus(const COandXTile* aControl) const;

private:
  RPointerArray<COandXTile> iTiles; // View owns the tiles
  COandXStatusWin* iStatusWin;      // and its own status window.
  TRect iBoardRect;  // Board area
  TRect iBorderRect; // Bounding rectangle for border
  TInt iTileSide;    // Tile dimension, allowing for line widths
                     // and border
  };
```

Note that the functions are declared public in the interface class, but private within `COandXAppView`. By providing each tile with an `MViewCmdHandler*` pointer to the enclosing view, we expose only those two functions (they are not even available via a `COandXAppView*` pointer).

The interface class is a useful way of encapsulating the interaction between the control and the view or, indeed, the rest of the program. The control does not care about any of the view's internal details, provided that it correctly handles the request to play in a particular square and the querying of a square's content. Similarly, the view does not care whether the command originated with a key event, a pointer event, or some other kind of event – it just requires these functions to be called at the right time with the right parameters.

This is a useful technique, which is used frequently within a variety of Symbian OS APIs, and it is a good one to adopt for your programs, especially if they are more complicated than the Noughts and Crosses application. As a further illustration, think about the way commands

reach `HandleCommandL()`. The menu bar uses an `MEikMenuObserver` for this and does not otherwise care about the large API of the `CEikAppUi` class. The button bar uses the `MEikCommandObserver` interface. Similarly, command buttons use an `MCoeControlObserver` interface, which the button bar implements by converting button events into application commands.

`TryHitSquareL()` simply calls the controller's `HitSquareL()` function, to determine whether the attempt is successful:

```
TBool COandXAppView::TryHitSquareL(const COandXTile* aControl)
  {
  return Controller().HitSquareL(Index(aControl));
  }
```

It returns `ETrue` only if the square was previously empty and now contains a symbol, otherwise it returns `EFalse`.

As was mentioned earlier, `TileStatus()` is called from each tile when it needs to draw itself, in order to determine which, if any, symbol should be drawn in the tile. The function's action is simply to request the information from the engine:

```
TTileState COandXAppView::TileStatus (const COandXTile* aControl) const
  {
  return Engine().TileStatus(Index(aControl));
  }
```

The majority of the view's functions are described in Chapter 15, where they are used to illustrate the general behavior of controls. Here, we mention only `ShowTurn()`, which is called from the application UI class and the controller whenever there is a change in whose turn it is to play. It simply requests the status window to redraw itself:

```
void COandXAppView::ShowTurn()
  {
  iStatusWin->DrawDeferred();
  }
```

How `DrawDeferred()` works is explained in Chapter 17; here it is sufficient to know that it results in a call to the status window's `Draw()` function. As we saw earlier, there is no need for `ShowTurn()` to pass a parameter indicating whose turn it is, since the status window's `Draw()` function queries the controller directly for the symbol it needs to draw.

Command Menus

The application uses only two commands: one to specify whether the Nought player or the Cross player should make the first move, and one

FIFTH THIRD EX1062
PART 2 of 3
Page 128 of 379

to abandon the current game. The first of these is only relevant before a game starts and the second is only used while a game is in progress. In consequence, we've chosen to use a dynamic command menu, which shows only those commands that can sensibly be used, given the current state of the game. We've implemented the menu in a way that illustrates some of the options that are available for setting up and changing the content of a command menu. Although at least one of the commands, that to select the first player, could reasonably be implemented via a dialog, we've chosen not to do so, leaving the discussion of dialog use to Chapter 16.

For S60, the specification of the command menu starts in the resource script, with the definition of an `EIK_APP_INFO` resource:

```
RESOURCE EIK_APP_INFO
  {
  menubar = r_oandx_menubar;
  cba = R_AVKON_SOFTKEYS_OPTIONS_EXIT;
  }
```

This resource contains a pointer to the resource that contains the application's menu bar and a Control Button Array that defines the softkey labels – in this case, Options and Exit – that are to appear in the control pane.

The menu bar resource, in turn, specifies one or more `MENU_PANE` resources that list the menu items they contain. As in this case, an S60 menu bar normally contains only one menu pane. If you specify more than one menu pane, the S60 UI concatenates their contents into a single list.

```
RESOURCE MENU_BAR r_oandx_menubar
  {
  titles =
    {
    MENU_TITLE
      {
      menu_pane = r_oandx_menu;
      }
    };
  }
```

For this application, the menu pane specifies only one menu item; the second menu item is constructed dynamically.

```
RESOURCE MENU_PANE r_oandx_menu
  {
  items =
    {
    MENU_ITEM
```

```
      {
      command = EOandXNewGame;
      txt = "New game";
      }
    };
  }
```

Each menu item specifies the text to be displayed and a non-zero value, unique within the application, used to identify the command. The command IDs are normally defined via an enumeration in the application's HRH file; for this application, they are defined, in `oandx.hrh`, as:

```
enum TOandXIds
  {
  EOandXNewGame = 0x1000,  // start value must not be 0
  EOandXSwitchTurn
  };
```

The resource file specifies only the command ID and text for the New game command. The other command is added dynamically according to context, using one or other of the following two resources for its text:

```
RESOURCE TBUF r_oandx_o_moves_first
  {
  buf = "Noughts move first";
  }

RESOURCE TBUF r_oandx_x_moves_first
  {
  buf = "Crosses move first";
  }
```

It is worth noting here that a truly commercial application would read the text for these resources – and most other text within the resource script – from a separate file, in order to facilitate translation into other languages. We have not done so in order to make the meaning of these resources clearer in the text of this book.

S60 applications that need to modify the content of their command menus should supply an implementation of the `DynInitMenuPaneL()` function of the application UI class, which is called each time a menu is about to be displayed. The `aResourceId` parameter identifies the menu and `aMenuPane` points to a menu pane that has already been constructed from the relevant resource. For this application, the implementation is:

```
void COandXAppUi::DynInitMenuPaneL(TInt aResourceId,
                            CEikMenuPane* aMenuPane)
  {
  if (aResourceId == R_OANDX_MENU)
```

```
    {
    if (iController->IsNewGame())
      {
      CEikMenuPaneItem::SData item;
      iCoeEnv->ReadResource(item.iText,
          iController-IsCrossTurn() ? R_OANDX_O_MOVES_FIRST :
                                      R_OANDX_X_MOVES_FIRST);
      item.iCommandId = EOandXSwitchTurn;
      item.iFlags = 0;
      item.iCascadeId = 0;
      aMenuPane->AddMenuItemL(item);

      aMenuPane->DeleteMenuItem(EOandXNewGame);
      }
    }
  }
```

The function first checks that the resource ID is for the relevant menu, but does not modify the menu pane (which therefore displays the New game menu item) if a game is in progress. Otherwise, a new menu item is constructed, in a `CEikMenuPaneItem::SData` data structure, using the relevant resource text, and added to the menu pane, and the New game item is deleted.

When the user selects a command, the S60 UI responds by calling the `HandleCommandL()` function of the application UI class. This is normally implemented via a `switch` statement, with each case corresponding to one of the possible command IDs:

```
void COandXAppUi::HandleCommandL(TInt aCommand)
  {
  switch(aCommand)
    {
    case EEikCmdExit:
    case EAknSoftkeyExit:
      {
      SaveL();
      Exit();
      }
      break;
    case EOandXNewGame:
      {
      iController->Reset();
      iAppView->ResetView();
      }
      break;
    case EOandXSwitchTurn:
      {
      iController->SwitchTurn();
      }
      break;
    default:
      break;
    }
  }
```

In the Noughts and Crosses application, each case, with the exception of the Exit command, corresponds to one of the command IDs specified in `oandx.hrh` and is handled simply by making the appropriate changes to the controller's state.

Depending on how the Exit command is initiated, it may have either of the standard command IDs shown in the above code. The application's response must always be to call `Exit()` but, in addition, it calls `SaveL()` to save the current game state in the application's persistent data.

12.2 Differences for UIQ 3

One of the major influences in the development of UIQ 3 was the need to make it easier for developers to support multiple form factors (both in terms of screen dimensions and use of portrait or landscape alignments) and a variety of input methods (touch screen or keyboard; menu-driven or softkey-driven interaction) from a single code base.

In order to facilitate such flexibility, all UIQ 3 applications need to derive their views from the new `CQikViewBase` or `CQikMultiPageViewBase` classes. The UIQ version of the Noughts and Crosses application derives its main view from the `CQikViewBase` view class (which is a subclass of `CCoeControl`) and not directly from `CCoeControl` itself. However, since it only has one view, it makes no specific use of the view architecture that is described in Chapter 14.

The Application UI Class

There is a small difference in the way that the application UI class creates and initializes the View. Remember that the S60 `ConstructL()` function of the application UI class is implemented as:

```
void COandXAppUi::ConstructL()
  {
  BaseConstructL(EAknEnableSkin);
  iEngine = COandXEngine::NewL();
  iController = COandXController::NewL();
  iAppView = COandXAppView::NewL(ClientRect());
  AddToStackL(iAppView); // Enable keypresses to the view
  iStacked = ETrue;
  ReportWhoseTurn();
  }
```

The destructor needed to take note of `iStacked` in order to determine whether or not it should remove the view from the control stack. For UIQ, the corresponding code is:

```
void COandXAppUi::ConstructL()
  {
```

```
  CQikAppUi::ConstructL(); // initiate the standard values
  iEngine = COandXEngine::NewL();
  iController = COandXController::NewL();
  // Create the view and add it to the framework
  iAppView = COandXView::NewL(*this);
  AddViewL(*iAppView);

  ReportWhoseTurn();
  }
```

The most significant difference is that the UIQ version registers the view by calling `AddViewL()`, which also takes care of adding it to the control stack. After this call, the application UI class has full ownership of the view, so you do not have to worry about either deregistering the view or removing it from the control stack when the application UI class is destroyed.

The View Class

The majority of the remaining source code differences that are needed in a UIQ version of the application stem from the differences between `CQikViewBase` and `CCoeControl`. So, while both forms of the view own their component controls, the mechanisms of ownership and access are different. A view that derives from `CCoeControl`, as in the S60 version, creates its component controls (normally in its `ConstructL()` function) and stores pointers to them in specific items of member data. In the S60 version, the relevant `ConstructL()` code is:

```
...
for (TInt i = 0; i < KNumberOfTiles; i++)
  {
  User::LeaveIfError(iTiles.Append(CreateTileL()));
  }
...
iStatusWin = COandXStatusWin::NewL(Window());
```

The controls are normally explicitly deleted in the destructor:

```
COandXAppView::~COandXAppView()
  {
  for (TInt i=0; i<KNumberOfTiles; i++)
    {
    delete iTiles[i];
    }
  iTiles.Close();
  delete iStatusWin;
  }
```

As described more fully in Chapter 17, access to each child control is via the `CountComponentControls()` and `ComponentControl()` functions that each component-owning control must implement:

```
TInt COandXAppView::CountComponentControls() const
  {
  return KNumberOfTiles + 1;
  }

CCoeControl* COandXAppView::ComponentControl(TInt aIndex) const
  {
  if (aIndex==KNumberOfTiles)
    {
    return iStatusWin;
    }
  else
    {
    return const_cast<COandXTile*>(iTiles[aIndex]);
    }
  }
```

`CQikViewBase` provides its own mechanism for component control ownership, so views that derive from this class do not need to provide explicit data members to store pointers to the components, nor do they need to provide code in the destructor to delete them. In addition, such views do not need to (and should not) provide implementations of `CountComponentControls()` and `ComponentControl()`.

Instead, UIQ views add component controls – normally from within the view's implementation of `ViewConstructL()` – to a built-in array that is initialized by a call to `InitComponentArrayL()` and then accessed via a `Components()` function, for example:

```
void COandXView::ViewConstructL()
  {
  ViewConstructFromResourceL(R_OANDX_UI_CONFIGURATIONS);
  InitComponentArrayL();
  for (TInt i = 0; i < KNumberOfTiles; i++)
    {
    COandXTile * tile = new(ELeave) COandXTile;
    AddControlLC(tile, i);
    tile->ConstructL(this);
    tile->SetFocusing(ETrue);
    CleanupStack::Pop(tile);
    }
  //  Status window added, but not created here
  AddControlLC(iStatusWin, KNumberOfTiles);
  CleanupStack::Pop(iStatusWin);
  iStatWinNotAppended = EFalse;
  // Status window now owned by the view

  UpdateCommandsL(Controller().IsNewGame(), Controller().IsCrossTurn());
  SetFocusByIdL(0);
  }
```

In the UIQ version of the application, as you can see from the above code, each tile is created and then appended to the array with a call to `AppendLC()` which, as its name implies, leaves a pointer to the

tile on the cleanup stack. The tile is removed from the cleanup stack once any potentially leaving initialization is complete. The view takes ownership of all component controls that are added in this way and they are automatically deleted when the view is destroyed.

> If you think carefully about the code sequence for adding a component into a view, you might begin to wonder if it is truly safe. For example, what if the actions performed by `AddControlLC()` cause a leave before the control is added to the cleanup stack? Furthermore, if the view takes ownership of the control, should `AddControlLC()` leave the control on the cleanup stack at all? Could a leave before the control is popped from the cleanup stack result in the control being deleted twice – once from the cleanup stack and once by the owning view? In Symbian OS, it is worth asking questions like this, since leave-safeness is paramount.
>
> In fact, this sequence is safe. `AddControlLC()` adds the control to the cleanup stack before performing any potentially leaving actions, and the item that is added to the cleanup stack is one that is specifically designed to transfer ownership of the view (in a leave-safe way) as it is popped.

The creation of the status window follows a slightly different pattern. The reason for this is that the status window is accessed from the controller, via the `ReportWhoseTurn()` function of the application UI class, and the first access occurs before the view's `ViewConstructL()` has been called. (To optimize start-up time, particularly for applications with multiple views, `ViewConstructL()` is only called at the time that the view first needs to be made visible.) To prevent this attempted access from causing a panic, we need to create the status window in the view's `ConstructL()` and add it in `ViewConstructL()`.

However, there is then a risk that the status window will not be deleted if the code leaves between its creation and it being successfully added. We solved this problem by including the `iStatWinNotAppended` flag in the class data, setting it when the status window is created and clearing it when the window has successfully been added to the array. The view's destructor has to delete the status window only if the flag is set:

```
COandXView::~COandXView()
  {
  if (iStatWinNotAppended)
  // Once appended, system code takes care of a component's deletion
    {
    delete iStatusWin;
    }
  }
```

It is also worth pointing out that the initial construction of the view, from `ViewConstructL()`, is by means of a call to `ViewConstructFromResourceL()`. In suitable cases, an entire multi-page view can be constructed from the resource accessed by this function, in the same way that a dialog can be constructed from a resource, as described in Chapter 16. In the present case, the resource specifies a single page, without supplying any content:

```
RESOURCE QIK_VIEW_PAGES r_oandx_layout_pages
  {
  pages =
    {
    QIK_VIEW_PAGE
      {
      page_id = EOandXViewPage;
      }
    };
  }
```

Since the UIQ version of the view class does not have accessible implementations of `ComponentControl()` or `CountComponentControls()`, functions that, in an S60 application, call one or both of these functions need a different implementation. In the Noughts and Crosses application, the functions that are affected are those associated with which tile has focus. In the S60 version, these are `IdOfFocusControl()` and `MoveFocusTo()`:

```
TInt COandXAppView::IdOfFocusControl()
  {
  TInt ret = -1;
  for (TInt i=0; i<KNumberOfTiles; i++)
    {
    if (ComponentControl(i)->IsFocused())
      {
      ret = i;
      break;
      }
    }
  ASSERT_ALWAYS(ret>=0, Panic(EOandXNoTileWithFocus));
  return ret;
  }

void COandXAppView::MoveFocusTo(const TInt index)
  {
  TInt oldIndex = IdOfFocusControl();
    if (index != oldIndex)
    {
    SwitchFocus(oldIndex, ComponentControl(index));
    }
  }
```

`SwitchFocus()` is implemented as:

```
void COandXAppView::SwitchFocus(TInt aFromIndex, CCoeControl* aToControl)
  {
  ComponentControl(aFromIndex)->SetFocus(EFalse, EDrawNow);
  aToControl->SetFocus(ETrue, EDrawNow);
  }
```

In the UIQ version, the two corresponding functions are `IdOfFocusedControl()` and `SetFocusByIdL()`, whose basic implementations are:

```
TInt COandXView::IdOfFocusedControl()
  {
  TInt i;
  for (i=0; i<KNumberOfTiles; i++)
    {
    CCoeControl *control=ControlById<CCoeControl>(i);
    if (control->IsFocused())
      break;
    }
  return i;
  }

void COandXView::SetFocusByIdL(TInt aControlId)
  {
  RequestFocusL(ControlById<CCoeControl>(aControlId));
  }
```

In the UIQ version, we changed the names of both functions to emphasize the differences in implementation. Note, in particular, that `SetFocusByIdL()` is a leaving function, whereas `MoveFocusTo()` is not. Fortunately this does not give rise to any other complications in the current application, since they are called only from within leaving functions.

While on the subject of focus, there are two further differences that need mentioning. First, the default status of newly created controls in S60 is for them to be capable of accepting focus, which is not the case in UIQ. In UIQ you need to call `SetFocusing(ETrue)` on each control that needs to accept focus (in this case, on each of the board's tiles). In contrast, in S60, you need to call `SetFocusing(EFalse)` on each control, such as the game's status window, that you do not want to be able to accept focus. Secondly, UIQ 3 views provide automatic, intelligent key- or button-based navigation between any of the component controls that can accept focus – that is, those controls for which `SetFocusing(ETrue)` has been called. In consequence, the UIQ version has no need to implement the view's `OfferKeyEventL()` function.

Commands

UIQ 3 views are primarily designed for use in applications with multiple views, each with its own set of commands, so they are not seen at their

best in an application such as the Noughts and Crosses application, which only has a single view. The application does, however, allow the basic differences to be illustrated.

A major difference in UIQ is that commands are treated in a more abstract way than in S60. A UIQ device may be set up to access its commands by means of either a pointer device or softkeys, and the commands themselves may or may not be displayed in a command menu. In addition, commands may be provided from a variety of sources within the application, such as one or more individual controls – or even from outside the application itself, for example, from a front-end processor.

The UIQ application framework contains a command model that deals with all these variations. In consequence, the application programmer need only be concerned with which commands should be available and not with how they are presented or selected.

The first change is to the resource script. As we saw earlier, an S60 application uses an `EIK_APP_INFO` resource, to specify the menu bar and the menu items it contains. In contrast, a UIQ 3 application must specify an empty `EIK_APP_INFO` resource, plus a `QIK_VIEW_CONFIGURATIONS` resource that lists the UI configurations it supports. In the Noughts and Crosses application, these resources are:

```
RESOURCE EIK_APP_INFO { }

RESOURCE QIK_VIEW_CONFIGURATIONS r_oandx_ui_configurations
  {
  configurations =
    {
    QIK_VIEW_CONFIGURATION
      {
      ui_config_mode = KQikPenStyleTouchPortrait;
      command_list = r_oandx_portrait_commands;
      view = r_oandx_layout;
      },
    QIK_VIEW_CONFIGURATION
    {
      ui_config_mode = KQikSoftkeyStylePortrait;
      command_list = r_oandx_portrait_commands;
      view = r_oandx_layout;
      }
    };
  }
```

Each configuration specifies its list of available commands via a `QIK_COMMAND_LIST` resource:

```
RESOURCE QIK_COMMAND_LIST r_oandx_portrait_commands
  {
  items =
    {
```

```
    QIK_COMMAND
      {
      id = EEikCmdExit;
      type = EQikCommandTypeScreen; // Only visible in debug
      stateFlags = EQikCmdFlagDebugOnly;
      text = "Close (debug)";
      },
    QIK_COMMAND
      {
      id = EOandXNewGame;
      type = EQikCommandTypeScreen;
      text = "New game";
      },
    QIK_COMMAND
      {
      id = EOandXFirstPlayer;
      type = EQikCommandTypeScreen;
      text = "Noughts move first";
      }
    };
  }
```

As in the S60 version, the commands are updated dynamically, but the mechanism is different. S60 applications update their menus via the `DynInitMenuPaneL()` function, which is called immediately before a menu is made visible.

In UIQ, since a command may be selected by means of a softkey, or from a toolbar, there is no guarantee that a menu pane is displayed before the selection takes place. In consequence, you cannot use `DynInitMenuPaneL()` to update the available commands. Indeed, since some commands may be permanently visible, UIQ applications need to perform such updating each time the view's state changes in a way that affects the commands.

In many cases, it would be appropriate to implement the view's `HandleControlEventL()` function, which is called (with an `EEventStateChanged` event code) each time one of the view's controls changes its state. In the Noughts and Crosses application, all significant state changes take place in the Controller, so we've taken the more direct option of adding a `SetStateL()` function to the Controller, implemented as:

```
void COandXController::SetStateL(TState aState)
  {
  iState = aState;
  OandXAppUi()->UpdateCommandsL(IsNewGame(), IsCrossTurn());
  }
```

This calls an `UpdateCommandsL()` function in the application UI class, which simply calls a similarly named function in the view:

```
void COandXView::UpdateCommandsL(TBool aIsNew, TBool aIsCrossTurn)
  {
```

```
  iCommandManager.SetAvailable(*this, EOandXNewGame, !aIsNew);
  iCommandManager.SetAvailable(*this, EOandXFirstPlayer, aIsNew);
  iCommandManager.SetTextL(*this, EOandXFirstPlayer,
      aIsCrossTurn?R_OANDX_O_FIRST:R_OANDX_X_FIRST);
  }
```

The following points are relevant when updating the menus:

- While a game is in progress, there is no need to worry about whether the text for the 'Who moves first' command is updated correctly or not, because that command is only available (and visible) before a new game is started.
- When executing the 'Who moves first' command, the game is always new, so the only interest is in setting the correct text for the next potential use of that command.

In addition, the controller's `Reset()` function is renamed to `ResetL()` because it can now leave:

```
void COandXController::ResetL()
  {
  Engine().Reset();
  if (IsCrossTurn())
    {
    SwitchTurn();
    }
  SetStateL(ENewGame);
  }
```

`ResetL()` cannot be used to initialize the controller (and engine) because `SetStateL()` calls the view's `UpdateCommandsL()` function, and the view does not exist at the time the controller is constructed. Instead, in the controller's `ConstructL()`, we explicitly set the relevant data:

```
...
Engine().Reset();
iState = ENewGame;
iCrossTurn = EFalse;
...
```

Since UIQ expects an application to have multiple views and each view may have a different set of commands, command processing is implemented within each individual view. A UIQ application, therefore, does not supply a `HandleCommandL()` function in the application UI class, but implements this function in each of its views. Note that the argument to this function is a reference to an instance of `CQikCommand`, rather than the command ID that is passed to this function in the

application UI class. Otherwise, the implementation is broadly similar to that in the S60 version:

```
void COandXView::HandleCommandL(CQikCommand& aCommand)
  {
  switch(aCommand.Id())
    {
    case EOandXNewGame:
      Controller().ResetL();
      SetFocusByIdL(0);
      DrawNow();
      break;
    case EOandXFirstPlayer:
      Controller().SwitchTurn();
      UpdateCommandsL(Controller().IsNewGame(),
                  Controller().IsCrossTurn());
      break;
    default:
#ifdef _DEBUG
      OandXAppUi()->SaveGameStateL();
#endif
      // The Go back and Exit commands are handled by CQikViewBase.
      CQikViewBase::HandleCommandL(aCommand);
      break;
      }
  }
```

Persistence

In UIQ, the default behavior is to allow applications to use persistent data, so, unlike in S60, there is no need to supply an implementation of the document's `OpenFileL()` function.

UIQ applications do not normally have a Close command (except for testing purposes, in debug builds). Instead, system code may instruct an application to close at any time that memory needs to be freed, first giving it an opportunity to save its persistent data. In order to access the functionality more easily, we have added – but only in debug builds – a call to the `SaveGameStateL()` function of the application UI class in the default case of the view's `HandleCommandL()`. The implementation is very simple:

```
void COandXAppUi::SaveGameStateL()
  {
  SaveL();
  }
```

We have to supply this new function because the `SaveL()` function of the application UI class is protected, and so cannot be called directly from the view. With this call in place, the persistent data is saved each time the Close (Debug) command is selected.

Since the application's persistent data contains one item – the ID of the tile that currently has focus – from the view, there is one final complication that we have to deal with. It is caused by the view's controls being created and initialized in the `ViewConstructL()` function which, as mentioned earlier, is called as late as possible, just before the view first becomes visible. This means that the document's `RestoreL()` function (and, at least when the application is run for the first time, its `StoreL()` function) are called before the view is fully initialized.

We resolve this problem by adding an `iInitialFocusId` data member to the `COandXView` class and coding the view's `SetFocusByIdL()` function as:

```
void COandXView::SetFocusByIdL(TInt aControlId)
  {
  CCoeControl * control = ControlById<CCoeControl>(aControlId);
  if (control)
    {
    RequestFocusL(control);
    }
  else // Control does not yet exist
    {
    iInitialFocusId = aControlId;
    }
  }
```

If the control does not yet exist, the value is copied into `iInitalFocusId` and used, within `ViewConstructL()`, to move focus to the correct control after it has been created.

We also need to deal with the case where the persistent data needs to be written before the tiles exist (this only happens once, the first time the application is run, when the application's INI file is being created). The relevant data is obtained by a call to the view's `IdOfFocusedControl()`, whose implementation is:

```
TInt COandXView::IdOfFocusedControl()
  {
  TInt i;
  for (i=0; i<KNumberOfTiles; i++)
    {
    CCoeControl *control=ControlById<CCoeControl>(i);
    if (!control) // Not yet created
      {
      return 0;
      }
    if (control->IsFocused())
      {
      break;
      }
    }
  return i;
  }
```

If a control does not yet exist, the function simply returns zero, which is the correct default value.

Summary

This chapter has described a simple but non-trivial application to play a game of Noughts and Crosses.

After a brief introduction to the game, the chapter explained the overall architecture of the application, before describing each of the S60 version's major component classes in more detail.

The chapter concluded with a discussion of the main differences between the S60 and UIQ versions of the game.

FIFTH THIRD EX1062
PART 2 of 3
Page 143 of 379

APPENDIX
HARRISON13

INTERNET ARCHIVE WEB TEXTS VIDEO AUDIO SOFTWARE All Search SIGN UP | LOG IN UPLOAD
ABOUT BLOG EVENTS PROJECTS HELP DONATE CONTACT JOBS VOLUNTEER

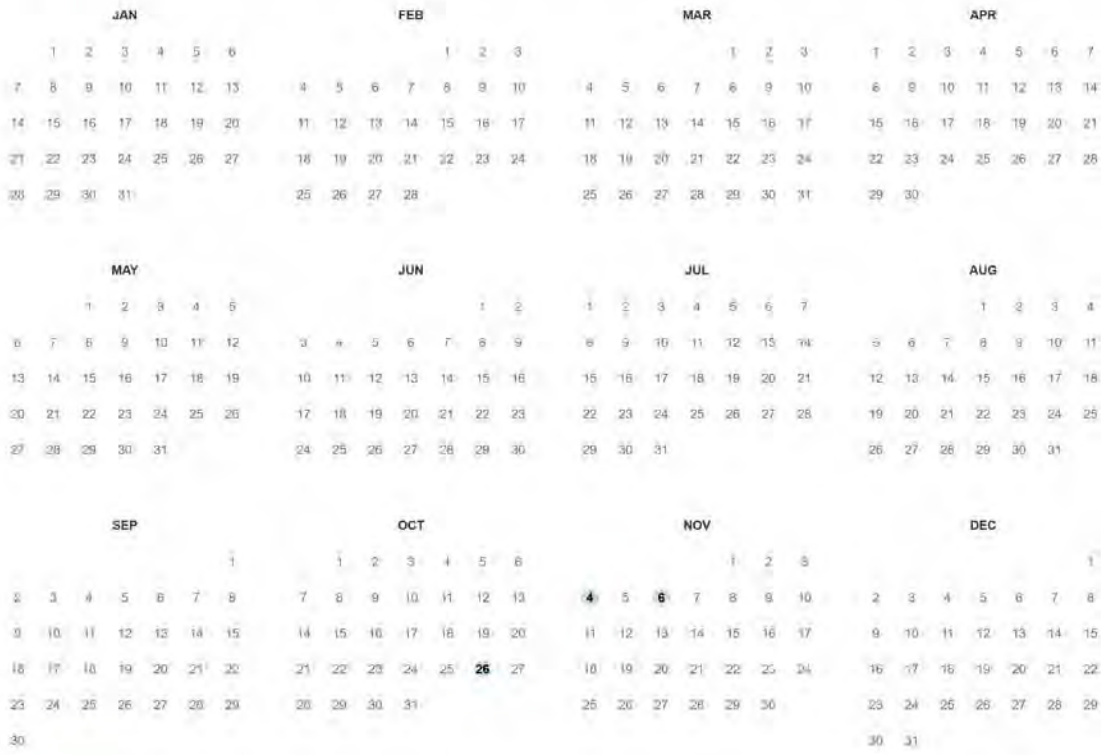
DONATE INTERNET ARCHIVE WaybackMachine Explore more than 1 trillion web pages saved over time

http://www.amazon.com/Symbian-OS-Mobile-Phones-Press/dp/0

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved **12 times** between October 26, 2007 and July 5, 2013.





Note

This calendar view maps the number of times **http://www.amazon.com/Symbian-OS-Mobile-Phones-Press/dp/0470066415** was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the FAQ.

FAQ | Contact Us | Terms of Service (Dec 31, 2014)

The Wayback Machine is an initiative of the Internet Archive, a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other projects include Open Library & archive-it.org.

Your use of the Wayback Machine is subject to the Internet Archive's Terms of Use

https://web.archive.org/web/20071026144849/http://www.amazon.com/Symbian-OS-Mobile-Phones-Press/dp/0470066415 July 30, 2026

http://www.amazon.com/Symbian-OS-Mobile-Phones-Press/dp/0470066415 Go SEP OCT JAN 26 2006 2007 2009 About this capture
12 captures 26 Oct 2007 - 5 Jul 2013



Share your own customer images
Search inside this book

Symbian OS C++ for Mobile Phones (Symbian Press) (Paperback)

by Richard Harrison (Editor), Mark Shackman (Editor)

(3 customer reviews)

List Price: $75.00
Price: **$75.00** & this item ships for **FREE with Super Saver Shipping**. Details

Availability: In Stock. Ships from and sold by **Amazon.com**. Gift-wrap available.

Want it delivered Monday, October 29? Choose **One-Day Shipping** at checkout. See details

Also Available in:	List Price:	Our Price:
Paperback (Pap/Cdr)	$65.00	$61.75

Quantity: 1
Add to Shopping Cart
or
Sign in to turn on 1-Click ordering.

More Buying Choices
Have one to sell? Sell yours here

Better Together

Buy this book with Symbian OS C++ for Mobile Phones: Programming with Extended Functionality and Advanced Features (Symbian Press) by Richard Harrison today!



+


Buy Together Today: $111.50
Buy both now!

Customers Who Bought This Item Also Bought


Symbian OS Explained: Effective C++ Programming for Smartphones (Symbian Press) by Jo Stichbury
(5) $51.00


Developing Series 60 Applications: A Guide for Symbian OS C++ Developers (Nokia Mobile Developer Series) by Leigh Edwards
(13) $34.64



Developing Software for Symbian OS: An Introduction to Creating Smartphone Applications in C++ (Symbian Press) by Steve Babin
(2) $70.00



Symbian OS Internals: Real-time Kernel Programming (Symbian Press) by Jane Sales
$60.08



Symbian OS Communications Programming (Symbian Press) by Iain Campbell
$60.00

› **Explore similar items:** Books (20)

Editorial Reviews

Book Description
Richard Harrison's existing books are the bestsellers in the Symbian Press Portfolio. His latest book, co-written with Mark Shackman is the successor to "Symbian OS C++ for Mobile Phones" Volumes One and Two. Written in the same style as the two previous volumes, this is set to be another gem in the series.

The existing material from the volumes will be combined, with explanations and example code updated to reflect the introduction of Symbian OS v9. New and simplified example application will be introduced, which will be used throughout the book. The reference and theory section in particular sets this book apart from the competition and complements other books being proposed at this time.

Anyone looking for a thorough insight into Symbian OS C++ before moving onto specialize on particular Symbian OS phones need this book! It will not teach people how to program in C++, but it will reinforce the techniques behind developing applications in Symbian OS C++, and more.

This innovative new book covers Symbian OS fundamentals, core concepts and UI.

Key highlights include:

- A quick guide to Kernel
- Platform security
- Publishing Applications
- View Architecture
- Multi-User games

Back Cover Copy
Programming Symbian OS is a key skill for mass market phone application development and this book is a must-have resource for any programmer planning to tool up to take advantage of the exciting opportunities offered by advanced mobile technology. *Symbian OS C++ for Mobile Phones* is based on the experience of Symbian engineers and will help you get to grips with all aspects of application development from basics to fully functioning complex applications.

This book provides you with the ultimate developer guide to Symbian OS C++ programming. The elegant, powerful architecture of Symbian OS is optimized for the mobile environment and for the demands of advanced communication on mobile networks. Whether you are developing applications and services for shipping mobile phones, or involved in pre-market mobile phone development, this book will help you understand the fundamental theory behind developing Symbian OS C++ code for constrained devices.

Symbian OS is the advanced, customizable operating system licensed by the worlds leading mobile phone manufacturers. It is designed for the specific requirements of advanced 2G, 2.5G and 3G mobile phones and includes a robust multitasking kernel, integrated telephony support, communications protocols, data management, advanced graphics support, a low level graphical user interface framework and a variety of application engines. *--This text refers to the **Paperback** edition.*

See all Editorial Reviews

Product Details

Paperback: 834 pages
Publisher: Wiley (August 24, 2007)

Language: English
ISBN-10: 0470066415
ISBN-13: 978-0470066416
Product Dimensions: 9.2 x 7.3 x 1.9 inches
Shipping Weight: 3.4 pounds (View shipping rates and policies)
Average Customer Review: based on 3 reviews.
Amazon.com Sales Rank: #184,674 in Books (See Bestsellers in Books)
Popular in these categories: (What's this?)
#8 in Books > Computers & Internet > Programming > APIs & Operating Environments > **EPOC-Symbian**
#15 in Books > Computers & Internet > Mobile & Wireless Computing > **Programming**
#38 in Books > Computers & Internet > Hardware > **Handheld & Mobile Devices**

(Publishers and authors: Improve Your Sales)
In-Print Editions: Paperback (Pap/Cdr) | All Editions

Would you like to **update product info** or **give feedback on images**? (We'll ask you to sign in so we can get back to you)

Inside This Book (learn more)

New!
Concordance | Text Stats

Browse Sample Pages:
Front Cover | Copyright | Table of Contents | Excerpt | Index | Surprise Me!

Search Inside This Book:
GO!

What Do Customers Ultimately Buy After Viewing Items Like This?



76% buy the item featured on this page:
Symbian OS C++ for Mobile Phones (Symbian Press)
$75.00



15% buy
Developing Software for Symbian OS: An Introduction to Creating Smartphone Applications in C++ (Symbian Press)
$70.00



9% buy
Developing Series 60 Applications: A Guide for Symbian OS C++ Developers (Nokia Mobile Developer Series)
$34.64

Compare these items **Explore similar items**

Tags customers associate with similar products (What's this?)

Click on a tag to find related items, discussions, and people.

windows mobile (7) | exam (2) | smartphone dev (4)
mobile (5) | exam prep (2) | usability (3)
mobile development (6) | programming (2) | microsoft (3)
pocket pc (6) | symbian (2)

Search Products Tagged with
GO!
See most popular Tags

Your tags: Add your first tag

Help others find this product - tag it for Amazon search
No one has tagged this product for Amazon search yet. Why not be the first to suggest a search for which it should appear?

Are you the publisher or author? Learn how Amazon can help you make this book an eBook.

If you are a publisher or author and hold the digital rights to a book, you can make it available as an eBook on Amazon.com. Learn more

Rate this item to improve your recommendations

I own it	Not rated	Your rating (Don't like it < > I love it!)						Save your rating
☐	?	1	2	3	4	5		Submit

Customer Reviews

3 Reviews
5 star: (0)
4 star: (0)
3 star: (2)
2 star: (1)
1 star: (0)

Average Customer Review
(3 customer reviews)
Create your own review

New! Amazon has customer video reviews
It's easy to shoot video reviews or life's everyday moments with the Flip Video camcorder.

Most Helpful Customer Reviews

Search Customer Reviews
GO!
Only search this product's reviews

15 of 17 people found the following review helpful:

Should've been titled - Programming the Symbian UIQ platform, June 20, 2003

By **"marcelol240"** (Florida, USA) - See all my reviews

This review is from: **Symbian OS C++ for Mobile Phones (Symbian Press) (Paperback)**

If you were looking for a deeper discussion of Symbian for Mobile Phones than the "Programming the Series 60 platform and Symbian OS", you've come to the wrong place. "Different" would be a more accurate word. Part of the problem is that it was put together by agglomerating works from various (18 !) authors. The writing style shifts from first to third person and back a lot.

The book rather briefly mentions Series 60 (still the predominant UI among symbian-based smartphones) and Series 80 (Nokia Communicators), and then moves on. All sample code provided by the authors (downloadable from the books website), is targeted strictly towards UIQ.

While the number of subjects covered(UI, file system, memory handling, Bluetooth, communications) serve their purpose, the book just seems too haphazardly composed to be easily digestable. At least they did provide a few small applets which demonstrate some of the subjects (strings, simple drawing) covered, before they present their version of the classic Battleship game (which in it's favor does implement a communication stack for multiplayer use

).

My ONE strong suggestion to the authors would've been not to ignore the Series 60 platform altogether. The last book for Series 60 left a lot of room for the authors of this book, to have tied the loose ends that the SDK leaves open, together.

5 of 5 people found the following review helpful:

Good Starting Point, December 30, 2003

By **J. Puopolo** (Sudbury, MA United States) - See all my reviews

This review is from: **Symbian OS C++ for Mobile Phones (Symbian Press) (Paperback)**

Having written software for 15 years for UNIX and Windows platforms, my move to the mobile space has been educational and humbling. Trying to learn to program for Symbian using the SDK and publicly available docs is daunting at best. The SDK docs are sparse, poorly written and provide a terrible search experience.

In contrast, Symbian OS C++ for Mobile Phones provides a solid foundation for learning to program this relatively new breed of devices. It covers a lot of material in a fair amount of depth and will help newbies to ascend the steep learning curve. Without this book, those new to Symbian programming will need to scour the SDK docs, trawl the newsgroups, and rely on kind souls from the UK to help with their inevitable programming questions.

I would have given this book 4 stars had it not been for several disappointing characteristics. First, there are too many authors. The writing style changes and information is repeated unnecessarily and in jarring ways. Second, the book sometimes repeats what's in the SDK docs, albeit in a consolidated form, without adding pragmatic insight and value. Third, it does not delve into some real-world, complex topics in any meaningful way. For example, the text quickly discusses messaging (SMS/MMS) but does not provide a digestible example of programmatically constructing an MMS. Fourth, it does not include Series 60 information, which is a shame given that the Series 60 is a very popular platform.

All in all, this is a good book that provides relevant, helpful information that will flatten the learning curve considerably.

A good base for UIQ, April 7, 2004

By **"chaussette"** (Paris, FRANCE) - See all my reviews

This review is from: **Symbian OS C++ for Mobile Phones (Symbian Press) (Paperback)**

I ordered this book because I was new to Symbian/UIQ. I found it easier and more convenient to use than the SDK on-line doc.
I appreciate the battleship case-study used to illustrate the basic and intermediate Symbian/UIQ concepts. It helped me a great deal. This book also focuses on UIQ application framework. But surprisingly, I found it very scarce (as much as the SDK) on some aspects of the application framework: embedded application programming, command-line parameters processing.
However still good to have it close at hand for when I need a quick info.

Share your thoughts with other customers: Create your own review

See all 3 customer reviews...

Customer Discussions Beta (What's this?)

New! Receive e-mail when new posts are made. Click the "Track it!" button on any discussion page.

This product's forum (0 discussions)

Discussion	Replies	Latest Post
No discussions yet		

Ask questions, Share opinions, Gain insight

Start a new discussion

Topic:

First post:

Receive e-mail when new posts are made

Start a new discussion Guidelines

Prompts for sign-in

Product Information from the Amapedia Community Beta (What's this?)

Be the first person to add an article about this item at Amapedia.com.

See featured Amapedia.com articles

Listmania!

Search Listmania! GO!

C++ Development on Symbian OS: A list by SkippyJon Jones "SkippyJon Jones"

Books I read in 2007: A list by Tiago Luchini

Learning J2ME programming: A list by Alejandro Woywood "alexos"

Create a Listmania! list

Look for Similar Items by Category

Computers & Internet > General
Computers & Internet > Hardware > Handheld & Mobile Devices
Computers & Internet > Microsoft > Development > C & C++ Windows Programming
Computers & Internet > Mobile & Wireless Computing > Programming
Computers & Internet > Networking > Networks, Protocols & APIs > General
Computers & Internet > Operating Systems > General
Computers & Internet > Programming > APIs & Operating Environments > EPOC-Symbian
Computers & Internet > Programming > C > General
Computers & Internet > Software > General
Qualifying Textbooks - Fall 2007 > All Titles

Look for Similar Items by Subject

- [ ] C & Visual C
- [ ] Operating systems & graphical user interfaces (GUIs)
- [ ] Computers
- [ ] Computers - Communications / Networking
- [ ] Computers - Languages / Programming
- [ ] Computer Books: General
- [ ] Networking - General
- [ ] Programming Languages - C++
- [ ] Computers / Networking / General
- [ ] Hardware - Handheld Computing

Find books matching ALL checked subjects
i.e., each book must be in subject 1 AND subject 2 AND ...

Great Deals on Magazines



Visit our huge selection of magazine subscriptions often to see the latest special offers and bonuses. Check out magazines like *The New Yorker*, *Wired*, and *Vanity Fair*.

Big Savings in Books

Bargain Books

Find great titles at fantastic prices in our Bargain Books Store.

New Low Prices on Garmin nüvis



As they prepare to roll out new models, Garmin has lowered prices on their entire line of slim nüvi 200s. Get them while they last.

› Shop all Garmin

Hot Deals for You

Amazon.com DELIVERS E-mail Subscriptions

Subscribe to the Camera News and Special Offers e-mail and get the latest Camera & Photo deals and new releases delivered right to your inbox once a month. Sign up today.

Where's My Stuff?
Track your recent orders.
View or change your orders in Your Account.

Shipping & Returns
See our shipping rates & policies.
Return an item (here's our Returns Policy).

Need Help?
Forgot your password? Click here.
Redeem or buy a gift certificate/card.
Visit our Help department.

Search Amazon.com GO!

amazon.com® Amazon.com Home | Directory of All Stores

International Sites: Canada | United Kingdom | Germany | Japan | France | China

Help | View Cart | Your Account | Sell Items | 1-Click Settings

Investor Relations | Press Release | Careers at Amazon | Join Associates | Join Advantage | Join Honor System | Advertise With Us

Conditions of Use | Privacy Notice © 1996-2007, Amazon.com, Inc. or its affiliates

APPENDIX
HARRISON14

https://web.archive.org/web/20070000000000*/http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470066415.html July 30, 2026

INTERNET ARCHIVE WEB TEXTS VIDEO AUDIO SOFTWARE All Search SIGN UP | LOG IN UPLOAD

ABOUT BLOG EVENTS PROJECTS HELP DONATE CONTACT JOBS VOLUNTEER

DONATE

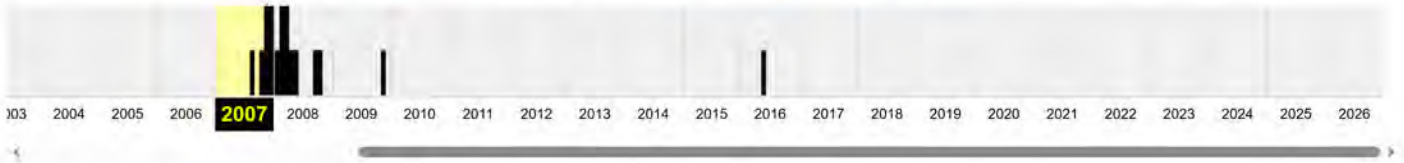
INTERNET ARCHIVE WaybackMachine

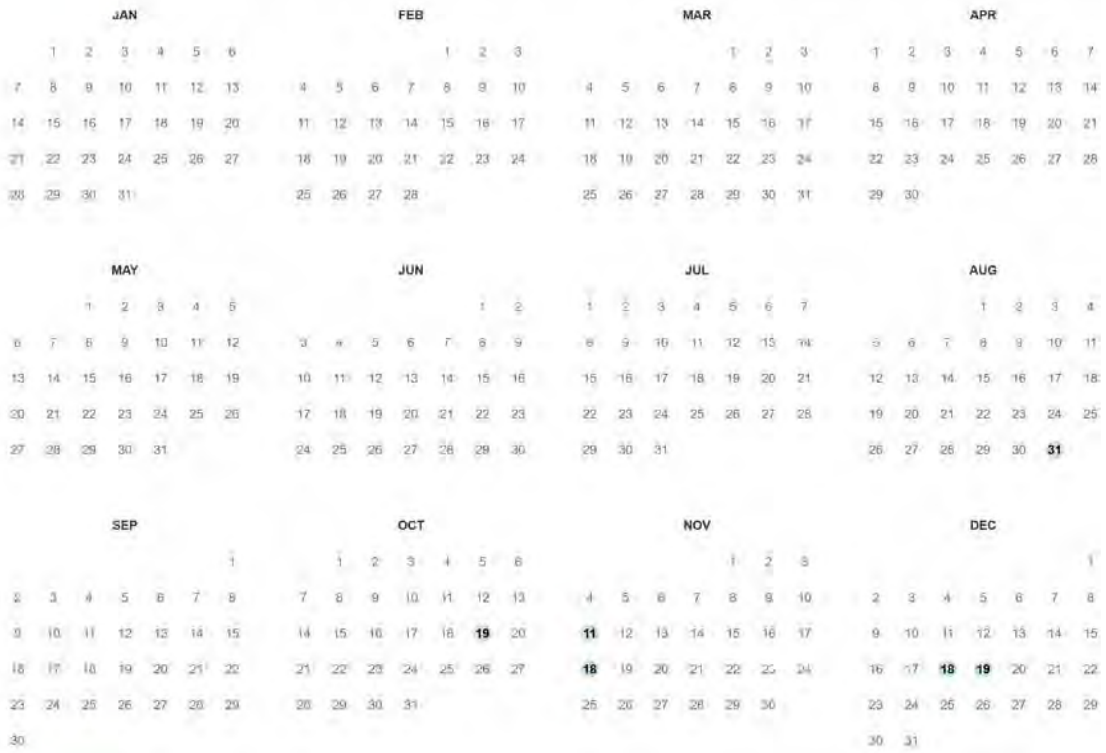
Explore more than 1 trillion web pages saved over time

http://www.wiley.com/WileyCDA/WileyTitle/productCd-04700664

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved **17 times** between August 31, 2007 and May 20, 2016.





Note

This calendar view maps the number of times **http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470066415.html** was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the FAQ.

FAQ | Contact Us | Terms of Service (Dec 31, 2014)

The Wayback Machine is an initiative of the Internet Archive, a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other projects include Open Library & archive-it.org.

Your use of the Wayback Machine is subject to the Internet Archive's Terms of Use

https://web.archive.org/web/20070831041448/http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470066415.html July 30, 2026

INTERNET ARCHIVE WayBackMachine http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470066415.html Go JUL AUG OCT 31 2006 2007 2008 About this capture
17 captures 31 Aug 2007 - 20 May 2016

Symbian OS C++ for Mobile Phones, Volume 3

SAVE View our Special Offers ›

RELATED SUBJECTS
- General Communication Technology
- Mobile & Wireless Communications
- Networks
- General Computer Engineering
- LINUX Networking
- Networking Security
- UNIX Networking

Join a Computing Mailing List. Exclusive offers, news and more.

RELATED TITLES

More By These Authors

Symbian OS C++ for Mobile Phones: Programming with Extended Functionality and Advanced Features, Volume 2

Symbian OS C++ for Mobile Phones, Volume One

General Networking

Teach Yourself VISUALLY Wireless Networking, 2nd Edition
by Rob Tidrow

Home Networking
by Sudhir Dixit (Editor), Ramjee Prasad (Editor)

Wireless Networking Visual Quick Tips
by Rob Tidrow

PC Technician Street Smarts: A Real World Guide to CompTIA A+ Skills
by James Pyles

Programming PC Connectivity Applications for Symbian OS: Smartphone Synchronization and Connectivity for Enterprise and Application Developers
by Ian McDowall


Cover image for product 0470066415

Larger Image

Symbian OS C++ for Mobile Phones, Volume 3

Richard Harrison (Editor), Mark Shackman (Editor)

ISBN: 978-0-470-06641-6
Paperback
834 pages
August 2007

US $75.00 Add to Cart

This price is valid for United States. Change location to view local pricing and availability.

Other Available Formats: Adobe E-Book

Read an Excerpt
- Read Excerpt: Chapter (PDF)
- Read Excerpt: Table of Contents (PDF)
- Read Excerpt: Index (PDF)

Download Acrobat

Description | Table of Contents | Author Information

Richard Harrison's existing books are the bestsellers in the Symbian Press Portfolio. His latest book, co-written with Mark Shackman is the successor to "Symbian OS C++ for Mobile Phones" Volumes One and Two. Written in the same style as the two previous volumes, this is set to be another gem in the series.

The existing material from the volumes will be combined, with explanations and example code updated to reflect the introduction of Symbian OS v9. New and simplified example application will be introduced, which will be used throughout the book. The reference and theory section in particular sets this book apart from the competition and complements other books being proposed at this time.

Anyone looking for a thorough insight into Symbian OS C++ before moving onto specialize on particular Symbian OS phones need this book! It will not teach people how to program in C++, but it will reinforce the techniques behind developing applications in Symbian OS C++, and more.

This innovative new book covers Symbian OS fundamentals, core concepts and UI.

Key highlights include:

- A quick guide to Kernel
- Platform security
- Publishing Applications
- View Architecture
- Multi-User games

Printer-ready version E-mail a friend

Copyright © 2000-2007 by John Wiley & Sons, Inc. or related companies. All rights reserved. Please read our Privacy Policy

APPENDIX
HARRISON15

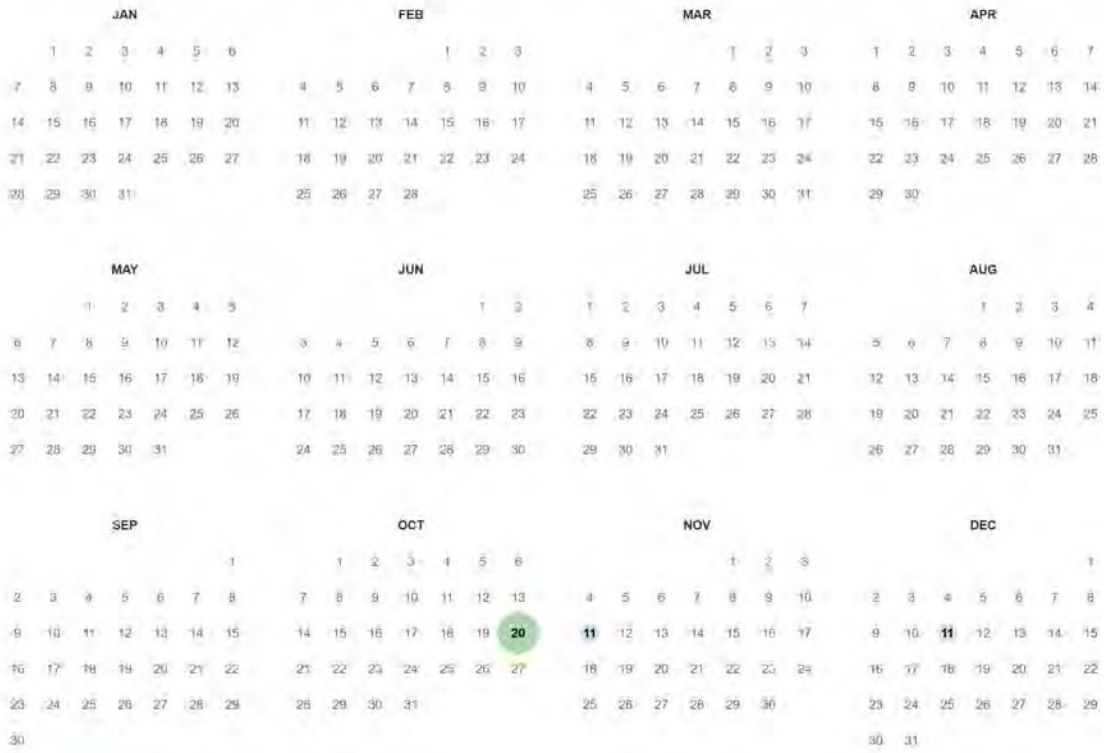
INTERNET ARCHIVE WEB TEXTS VIDEO AUDIO SOFTWARE All Search SIGN UP | LOG IN UPLOAD
ABOUT BLOG EVENTS PROJECTS HELP DONATE CONTACT JOBS VOLUNTEER

DONATE INTERNET ARCHIVE WaybackMachine Explore more than 1 trillion web pages saved over time
http://wiley.com/WileyCDA/WileyTitle/productCd-0470066415%2

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved **9 times** between October 20, 2007 and October 27, 2008.





Note

This calendar view maps the number of times **http://wiley.com/WileyCDA/WileyTitle/productCd-0470066415%2CdescCd-tableOfContents.html** was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the FAQ.
Green indicates redirects (3xx).

FAQ | Contact Us | Terms of Service (Dec 31, 2014)

The Wayback Machine is an initiative of the Internet Archive, a 501(c)(3) non-profit building a digital library of Internet sites and other cultural artifacts in digital form. Other projects include Open Library & archive-it.org

Your use of the Wayback Machine is subject to the Internet Archive's Terms of Use

https://web.archive.org/web/20071020033432/http://wiley.com/WileyCDA/WileyTitle/productCd-0470066415,descCd-tableOfContents.html July 30, 2026

INTERNET ARCHIVE WayBackMachine http://wiley.com/WileyCDA/WileyTitle/productCd-0470066415%2CdescCd-tableOfContents.html Go SEP OCT DEC 20 2006 2007 2008 About this capture
9 captures
20 Oct 2007 - 27 Oct 2008

Symbian OS C++ for Mobile Phones, Volume 3

SAVE View our Special Offers ›

RELATED SUBJECTS

- General Communication Technology
- Mobile & Wireless Communications
- Networks
- General Computer Engineering
- LINUX Networking
- Networking Security
- UNIX Networking

Join a Computing Mailing List. Exclusive offers, news and more.

RELATED TITLES

More By These Authors

Symbian OS C++ for Mobile Phones: Programming with Extended Functionality and Advanced Features, Volume 2

Symbian OS C++ for Mobile Phones, Volume One

General Networking

Teach Yourself VISUALLY Wireless Networking, 2nd Edition
by Rob Tidrow

Technologies for Home Networking
by Sudhir Dixit (Editor), Ramjee Prasad (Editor)

Wireless Networking Visual Quick Tips
by Rob Tidrow

PC Technician Street Smarts: A Real World Guide to CompTIA A+ Skills
by James Pyles

Programming PC Connectivity Applications for Symbian OS: Smartphone Synchronization and Connectivity for Enterprise and Application Developers
by Ian McDowall


Cover image for product 0470066415 Larger Image

Symbian OS C++ for Mobile Phones, Volume 3

Richard Harrison (Editor), Mark Shackman (Editor)

ISBN: 978-0-470-06641-6
Paperback
834 pages
August 2007

US $75.00 Add to Cart

This price is valid for United States. Change location to view local pricing and availability.

View Previous Edition of This Title
Other Available Formats: Adobe E-Book

Evaluation Copy

Instructors may request an evaluation copy for this title.

Read an Excerpt

- Read Excerpt: Chapter (PDF)
- Read Excerpt: Table of Contents (PDF)
- Read Excerpt: Index (PDF)

Download Acrobat

- Description
- Table of Contents
- Author Information

About the Authors.

Acknowledgements.

Symbian Press Acknowledgments.

About this book.

Glossary.

1 Getting Started.

1.1 Using the Emulator.

1.2 Hello World – Text Version.

Summary.

2 A System Introduction to Symbian OS.

2.1 Symbian OS Basics.

2.2 Framework Basics.

2.3 APIs Covered in this Book.

Summary.

3 Symbian OS C++.

3.1 Fundamental Data Types.

3.2 Naming Conventions.

3.3 Functions.

3.4 APIs.

3.5 Templates.

3.6 Casting.

3.7 Classes.

3.8 Design Patterns.

Summary.

4 Objects – Memory Management, Cleanup and Error Handling.

4.1 Object Creation and Destruction.

4.2 Class Categories in Symbian OS.

4.3 Error Handling.

4.4 The Cleanup Stack.

4.5 Two-Phase Construction.

Summary.

5 Descriptors.

5.1 Overview.

5.2 Anatomy of Descriptors.

5.3 Literals.

5.4 Stack Descriptors.

5.5 Pointer Descriptors.
5.6 Heap Descriptors.
5.7 Narrow, Wide and Neutral Descriptors.
5.8 Descriptors and Binary Data.
5.9 Using Descriptors with Methods.
5.10 Some Descriptor Operations.
5.11 Correct Use of Descriptors.
5.12 Manipulating Descriptors.
Summary.
6 Active Objects.
6.1 The Asynchronous Service.
6.2 Multitasking and Pre-emption.
6.3 A More In-depth Look at Active Objects.
6.4 How It Works.
6.5 Active Object Priorities.
6.6 Active Object Cancellation.
6.7 Starting and Stopping the Scheduler.
6.8 Understanding a Stray Signal.
6.9 Other Common Active Object Errors.
6.10 Implementing State Machines.
6.11 Long-Running Tasks and Active Objects.
Summary.
7 Files and the File System.
7.1 File-Based Applications.
7.2 Drives and File Types.
7.3 File System Services.
7.4 Streams.
7.5 Stores.
Summary.
8 Interprocess Communication Mechanisms.
8.1 Overview.
8.2 Client–server IPC.
8.3 Publish and Subscribe IPC.
8.4 Message Queue IPC.
8.5 Which IPC Mechanism Should You Use?
Summary.
9 Platform Security and Publishing Applications.
9.1 Releasing an Application.
9.2 How Does Platform Security Work?
9.3 How Do I Support Platform Security?
9.4 Preparing an Application for Distribution.
9.5 Overview of Symbian Signed.
9.5 Installing a SIS File.
9.6 List of Capabilities.
Summary.
10 Debugging and the Emulator.
10.1 Using the Emulator.
10.2 Emulator Debugging.
10.3 Debugging on a Phone.
10.4 Miscellaneous Tools.
Summary.
11 The Application Framework.
11.1 Symbian OS Application Framework.
11.2 S60 and UIQ Platform Application Frameworks.
11.3 A Graphical Hello World.
Summary.
12 A Simple Graphical Application.
12.1 Implementing the Game on S60.
12.2 Differences for UIQ 3.
Summary.
13 Resource Files.

13.1 Why a Symbian-Specific Resource Compiler?
13.2 Source File Syntax.
13.3 Bitmaps and Icons.
13.4 Updating the Resource Files.
13.5 Application Registration Files.
13.6 Localizable Strings.
13.7 Multiple Resource Files.
13.8 Compiling a Resource File.
13.9 The Content of a Compiled Resource File.
13.10 Reading Resource Files.
Summary.
14 Views and the View Architecture.
14.1 The View Architecture.
14.2 The MCoeView Interface.
14.3 Introduction to the Example Application.
14.4 Creating and Managing the Views.
14.5 Implementing the MCoeView Interface.
14.6 Command Menus.
Summary.
15 Controls.
15.1 What Is a Control?
15.2 Control Types.
15.3 Control Layout.
15.4 Handling Key and Pointer Events.
15.5 Observing a Control.
15.6 Drawing a Control.
15.7 Backed-up Windows.
15.8 Backed-up-Behind Windows.
15.9 Dimmed and Invisible Controls.
Summary.
16 Dialogs.
16.1 What Is a Dialog?
16.2 Simple Dialogs.
16.3 Complex Dialogs.
16.4 Single-Page Dialogs.
16.5 Multi-Page Dialogs.
16.6 Dialog APIs.
16.7 Stock Controls for Dialogs.
16.8 Custom Controls in Dialogs.
Summary.
17 Graphics for Display.
17.1 Drawing Basics.
17.2 The CGraphicsContext API.
17.3 Drawing and Redrawing.
17.4 Drawing Controls.
17.5 Sharing the Screen.
17.6 Support for Drawing in CCoeControl.
17.7 Special Effects.
17.8 Window Server Features.
17.9 Device- and Size-Independent Graphics.
Summary.
18 Graphics for Interaction.
18.1 Key, Pointer and Command Basics.
18.2 User Requirements for Interaction.
18.3 Some Basic Abstractions.
18.4 Processing Key Events.
18.5 Processing Pointer Events.
18.6 Window Server and Control Environment APIs.
Summary.
19 Plug-ins and Extensibility.
19.1 System Services.

19.2 What Is a Plug-in?
19.3 The ECOM Library.
19.4 Plug-ins in Symbian OS.
Summary.
20 Communications and Messaging Services.
20.1 Communications in Noughts and Crosses.
20.2 Communication Between Controller and Transport.
20.3 Serial Communications.
20.4 Socket-based Communications.
20.5 Messaging.
20.6 Security.
Summary.
21 Multimedia.
21.1 The Multimedia Framework.
21.2 The Image Conversion Library.
21.3 Camera API.
21.4 Tuner API.
Summary.
22 Introduction to SQL RDBMS.
22.1 Overview of RDBMS.
22.2 SQL Basics.
22.3 Symbian SQL Server Component Architecture.
22.4 Symbian SQL Error Codes.
Summary.
Appendix: Developer Resources.
References.
Index.

Printer-ready version E-mail a friend

Resources for Authors | Librarians | Instructors | Booksellers | Press | Investors
Copyright © 2000-2007 by John Wiley & Sons, Inc. or related companies. All rights reserved. Please read our Privacy Policy

Get RSS Feeds of New Wiley Titles Site Map

APPENDIX
HARRISON16

https://web.archive.org/web/20080000000000*/http://media.wiley.com/product_data/excerpt/15/04700664/0470066415-1.pdf July 30, 2026

INTERNET ARCHIVE WEB TEXTS VIDEO AUDIO SOFTWARE All Search SIGN UP | LOG IN UPLOAD

ABOUT BLOG EVENTS PROJECTS HELP DONATE CONTACT JOBS VOLUNTEER

DONATE

INTERNET ARCHIVE WaybackMachine

Explore more than 1 trillion web pages saved over time

http://media.wiley.com/product_data/excerpt/15/04700664/0470(

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved **1 time** September 5, 2008

003 2004 2005 2006 2007 **2008** 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026

JAN

	1	2	3	4	5	
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

FEB

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	

MAR

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

APR

		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

MAY

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

JUN

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

JUL

		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

AUG

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

SEP

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

OCT

			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

NOV

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

DEC

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Note

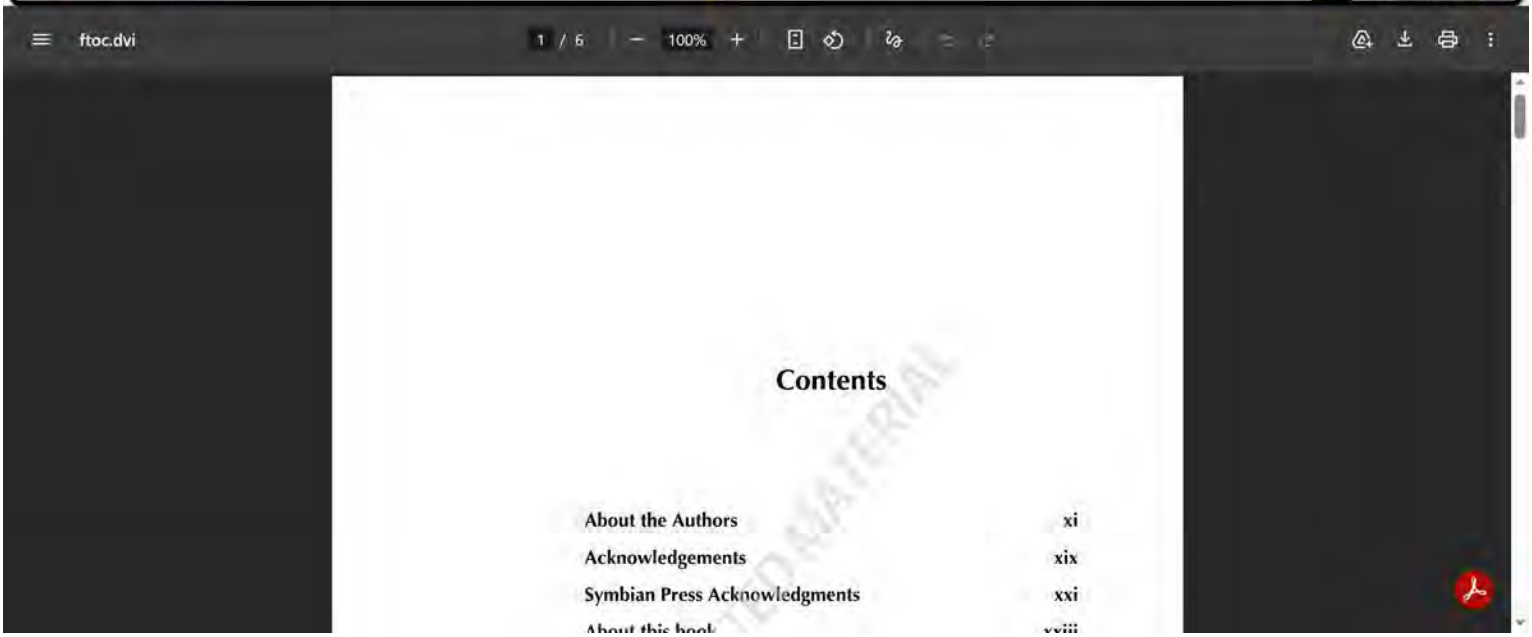
This calendar view maps the number of times **http://media.wiley.com/product_data/excerpt/15/04700664/0470066415-1.pdf** was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the FAQ.

FAQ | Contact Us | Terms of Service (Dec 31, 2014)

The Wayback Machine is an initiative of the Internet Archive, a 501(c)(3) non-profit building a digital library of Internet sites and other cultural artifacts in digital form. Other projects include Open Library & archive-it.org.

Your use of the Wayback Machine is subject to the Internet Archive's Terms of Use

https://web.archive.org/web/20080905091606/http://media.wiley.com/product_data/excerpt/15/04700664/0470066415-1.pdf July 30, 2026
http://media.wiley.com/product_data/excerpt/15/04700664/0470066415-1.pdf Go AUG SEP 05 OCT
1 capture 2007 2008
About this capture



Contents

About the Authors xi
Acknowledgements xix
Symbian Press Acknowledgments xxi
About this book xxiii

RESTRICTED MATERIAL

Contents

About the Authors **xi**

Acknowledgements **xix**

Symbian Press Acknowledgments **xxi**

About this book **xxiii**

Glossary **xxvii**

1 Getting Started **1**
1.1 Using the Emulator 1
1.2 Hello World – Text Version 6
Summary 15

2 A System Introduction to Symbian OS **17**
2.1 Symbian OS Basics 17
2.2 Framework Basics 30
2.3 APIs Covered in this Book 36
Summary 39

3 Symbian OS C++ **41**
3.1 Fundamental Data Types 41
3.2 Naming Conventions 43
3.3 Functions 49
3.4 APIs 51
3.5 Templates 55
3.6 Casting 56

COPYRIGHTED MATERIAL

3.7 Classes 57
3.8 Design Patterns 59
Summary 60

4 Objects – Memory Management, Cleanup and Error Handling 61
4.1 Object Creation and Destruction 62
4.2 Class Categories in Symbian OS 66
4.3 Error Handling 70
4.4 The Cleanup Stack 76
4.5 Two-Phase Construction 81
Summary 85

5 Descriptors 87
5.1 Overview 87
5.2 Anatomy of Descriptors 91
5.3 Literals 93
5.4 Stack Descriptors 93
5.5 Pointer Descriptors 95
5.6 Heap Descriptors 99
5.7 Narrow, Wide and Neutral Descriptors 108
5.8 Descriptors and Binary Data 109
5.9 Using Descriptors with Methods 110
5.10 Some Descriptor Operations 121
5.11 Correct Use of Descriptors 128
5.12 Manipulating Descriptors 138
Summary 148

6 Active Objects 151
6.1 The Asynchronous Service 151
6.2 Multitasking and Pre-emption 156
6.3 A More In-depth Look at Active Objects 158
6.4 How It Works 167
6.5 Active Object Priorities 171
6.6 Active Object Cancellation 176
6.7 Starting and Stopping the Scheduler 181
6.8 Understanding a Stray Signal 182
6.9 Other Common Active Object Errors 187
6.10 Implementing State Machines 190
6.11 Long-Running Tasks and Active Objects 194
Summary 200

7 Files and the File System 203
7.1 File-Based Applications 203
7.2 Drives and File Types 204

7.3 File System Services 205
7.4 Streams 216
7.5 Stores 224
Summary 239

8 Interprocess Communication Mechanisms 241
8.1 Overview 241
8.2 Client–server IPC 247
8.3 Publish and Subscribe IPC 253
8.4 Message Queue IPC 257
8.5 Which IPC Mechanism Should You Use? 259
Summary 261

9 Platform Security and Publishing Applications 263
9.1 Releasing an Application 263
9.2 How Does Platform Security Work? 264
9.3 How Do I Support Platform Security? 266
9.4 Preparing an Application for Distribution 268
9.5 Overview of Symbian Signed 272
9.5 Installing a SIS File 279
9.6 List of Capabilities 280
Summary 281

10 Debugging and the Emulator 283
10.1 Using the Emulator 283
10.2 Emulator Debugging 289
10.3 Debugging on a Phone 306
10.4 Miscellaneous Tools 308
Summary 309

11 The Application Framework 311
11.1 Symbian OS Application Framework 311
11.2 S60 and UIQ Platform Application Frameworks 313
11.3 A Graphical Hello World 315
Summary 330

12 A Simple Graphical Application 331
12.1 Implementing the Game on S60 333
12.2 Differences for UIQ 3 355
Summary 366

13 Resource Files 367
13.1 Why a Symbian-Specific Resource Compiler? 367
13.2 Source File Syntax 368
13.3 Bitmaps and Icons 372

FIFTH THIRD EX1062
PART 2 of 3
Page 164 of 379

13.4 Updating the Resource Files 377
13.5 Application Registration Files 378
13.6 Localizable Strings 379
13.7 Multiple Resource Files 382
13.8 Compiling a Resource File 383
13.9 The Content of a Compiled Resource File 385
13.10 Reading Resource Files 388
Summary 389

14 Views and the View Architecture 391
14.1 The View Architecture 392
14.2 The `MCoeView` Interface 397
14.3 Introduction to the Example Application 398
14.4 Creating and Managing the Views 406
14.5 Implementing the `MCoeView` Interface 408
14.6 Command Menus 414
Summary 421

15 Controls 423
15.1 What Is a Control? 423
15.2 Control Types 424
15.3 Control Layout 429
15.4 Handling Key and Pointer Events 432
15.5 Observing a Control 442
15.6 Drawing a Control 444
15.7 Backed-up Windows 451
15.8 Backed-up-Behind Windows 452
15.9 Dimmed and Invisible Controls 454
Summary 455

16 Dialogs 457
16.1 What Is a Dialog? 457
16.2 Simple Dialogs 459
16.3 Complex Dialogs 462
16.4 Single-Page Dialogs 464
16.5 Multi-Page Dialogs 466
16.6 Dialog APIs 472
16.7 Stock Controls for Dialogs 475
16.8 Custom Controls in Dialogs 477
Summary 480

17 Graphics for Display 481
17.1 Drawing Basics 482
17.2 The `CGraphicsContext` API 486

17.3 Drawing and Redrawing 492
17.4 Drawing Controls 499
17.5 Sharing the Screen 502
17.6 Support for Drawing in `CCoeControl` 514
17.7 Special Effects 521
17.8 Window Server Features 526
17.9 Device- and Size-Independent Graphics 529
Summary 556

18 Graphics for Interaction 557
18.1 Key, Pointer and Command Basics 558
18.2 User Requirements for Interaction 559
18.3 Some Basic Abstractions 561
18.4 Processing Key Events 567
18.5 Processing Pointer Events 571
18.6 Window Server and Control Environment APIs 576
Summary 582

19 Plug-ins and Extensibility 583
19.1 System Services 583
19.2 What Is a Plug-in? 587
19.3 The ECOM Library 593
19.4 Plug-ins in Symbian OS 603
Summary 604

20 Communications and Messaging Services 605
20.1 Communications in Noughts and Crosses 605
20.2 Communication Between Controller and Transport 606
20.3 Serial Communications 609
20.4 Socket-based Communications 615
20.5 Messaging 632
20.6 Security 646
Summary 649

21 Multimedia 651
21.1 The Multimedia Framework 651
21.2 The Image Conversion Library 682
21.3 Camera API 699
21.4 Tuner API 706
Summary 712

22 Introduction to SQL RDBMS 715
22.1 Overview of RDBMS 715
22.2 SQL Basics 716

FIFTH THIRD EX1062
PART 2 of 3
Page 166 of 379

22.3 Symbian SQL Server Component Architecture 729
22.4 Symbian SQL Error Codes 751
Summary 753

Appendix: Developer Resources 755

References 761

Index 763

APPENDIX
HARRISON17

DC Field	Value	Language
dc.contributor.advisor	Miksch, Silvia	-
dc.contributor.author	Böhme, Andreas	-
dc.date.accessioned	2020-06-30T06:02:40Z	-
dc.date.issued	2007	-
dc.date.submitted	2007-12	-
dc.identifier.citation	Böhme, A. (2007). A generic software template to support learning on mobile devices [Master Thesis, Technische Universität Wien]. reposiTUm. https://resolver.obvsg.at/urn:nbn:at:at-ubtuw:1-20911	-
dc.identifier.uri	https://resolver.obvsg.at/urn:nbn:at:at-ubtuw:1-20911	-
dc.identifier.uri	http://hdl.handle.net/20.500.12708/10841	-
dc.description	Zsfassung in dt. Sprache	-
dc.description.abstract	Nowadays almost everybody owns a mobile phone and carries it around the whole day. It is obvious, though, to use it not only as a phone, but also as a tool to do other things, like reading news or watching videos while travelling from or to e.g. work or school. Cell phone manufacturers acted on that idea, by producing so called Smart Phones, that are equipped with e-mail clients, web-browsers and multimedia-players, beside other useful applications. Going a step further, this work presents the concept of learning with someone's mobile- or smart-phone. Based on state of the art research in the fields of mobile learning and game-based learning, I am presenting the implementation of a mobile game-based learning approach. It incorporates learning strategies with fun elements to make it more attractive to the target group of students at the age of 18-24 years. While describing research basics, as well as the whole system-architecture, the work focuses on the technical specification and implementation of the mobile application. 	de
dc.description.abstract	Nahezu jeder Mensch besitzt heutzutage ein Mobiltelefon, und hat es wohl den ganzen Tag bei sich. Was wäre naheliegender, es nicht nur als Telefon zu verwenden, sondern auch um unterwegs z.B. Nachrichten zu lesen oder Filme zu sehen. Hersteller von Handys kamen zur gleichen Schlussfolgerung, und begannen sogenannte Smart Phones herzustellen. Diese werden standardmäßig mit E-Mail Clients, Web-Browsern, Multimedia-Playern, sowie einigen anderen nützlichen Anwendungen ausgestattet. Auf dieser Idee aufbauend, stellt diese Arbeit ein Konzept zum Lernen unter Verwendung von Mobiltelefonen dar. Basierend auf aktuellen Forschungsergebnissen im Bereich des mobilen, sowie des Spiel-basierten Lernens, möchte ich hier einen Ansatz vorstellen, der die beiden Forschungsbereiche kombiniert. Es werden dabei Lernstrategien mit Spaßelementen verbunden, um vor allem die Zielgruppe von 18-24 jährigen StudentInnen anzusprechen. Die zugrunde liegenden Forschungsergebnisse, sowie die gesamte System-Architektur werden ausreichend dargestellt, das Hauptaugenmerk liegt jedoch auf der technischen Spezifizierung und Implementierung der mobilen Anwendung.	en
dc.language	English	-
dc.language.iso	en	-

Page view(s)
383
checked on Nov 23, 2023

Download(s)
182
checked on Nov 23, 2023

Google Scholar™
Check

dc.rights.uri	http://rightsstatements.org/vocab/InC/1.0/	-
dc.subject	Mobil	de
dc.subject	JavaME	de
dc.subject	mobiles Lernen	de
dc.subject	spiel-basiertes Lernen	de
dc.subject	MGBL	de
dc.subject	Ahead of the Game	de
dc.subject	Crisis!	de
dc.subject	Fastest First	de
dc.subject	Mobile	en
dc.subject	JavaME	en
dc.subject	Mobile Learning	en
dc.subject	Game-based Learning	en
dc.subject	MGBL	en
dc.subject	Ahead of the Game	en
dc.subject	Crisis!	en
dc.subject	Fastest First	en
dc.title	A generic software template to support learning on mobile devices	en
dc.type	Thesis	en
dc.type	Hochschulschrift	de
dc.rights.license	In Copyright	en
dc.rights.license	Urheberrechtsschutz	de
dc.contributor.affiliation	TU Wien, Österreich	-
dc.rights.holder	Andreas Böhme	-
tuw.version	vor	-
tuw.thesisinformation	Technische Universität Wien	-
tuw.publication.orgunit	E188 - Institut für Softwaretechnik und Interaktive Systeme	-
dc.type.qualificationlevel	Diploma	-
dc.identifier.libraryid	AC05036305	-
dc.description.numberofpages	90 Pages	-
dc.identifier.urn	urn:nbn:at:at-ubtuw:1-20911	-
dc.thesistype	Masterarbeit	de
dc.thesistype	Master Thesis	en
dc.rights.identifier	In Copyright	en
dc.rights.identifier	Urheberrechtsschutz	de
tuw.advisor.orcid	0000-0003-4427-5703	-
item.languageiso639-1	en	-
item.openairetype	master thesis	-
item.grantfulltext	open	-
item.fulltext	with Fulltext	-
item.cerifentitytype	Publications	-
item.mimetype	application/pdf	-
item.openairecristype	http://purl.org/coar/resource_type/c_bdcc	-
item.openaccessfulltext	Open Access	-

crisitem.author.dept	TU Wien	-
Appears in Collections:	Thesis	

Fulltext (Version of Record (published version)) In Copyright	Adobe PDF	(2.74 MB)

SHOW SIMPLE ITEM RECORD

© 2026 TU Wien

Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).



TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

MASTERARBEIT

A Generic Software Template to support Learning on Mobile Devices

Ausgeführt am Institut für
Softwaretechnik und interaktive Systeme
der Technischen Universität Wien
unter der Anleitung von
Ao.Univ.Prof. Mag. Dr. Silvia Miksch
durch

Bakk. techn. Andreas Böhme

Sonnleithnergasse 15/3/4/50
A-1100 Wien

Wien, am 18. Dezember 2007
Datum

Unterschrift (Student)

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Wien, 18. Dezember 2007

Unterschrift

FIFTH THIRD EX1062
PART 2 of 3
Page 173 of 379

Kurzfassung

Nahezu jeder Mensch besitzt heutzutage ein Mobiltelefon, und hat es wohl den ganzen Tag bei sich. Was wäre naheliegender, es nicht nur als Telefon zu verwenden, sondern auch um unterwegs z.B. Nachrichten zu lesen oder Filme zu sehen. Hersteller von Handys kamen zur gleichen Schlussfolgerung, und begannen sogenannte *Smart Phones* herzustellen. Diese werden standardmäßig mit E-Mail Clients, Web-Browsern, Multimedia-Playern, sowie einigen anderen nützlichen Anwendungen ausgestattet.

Auf dieser Idee aufbauend, stellt diese Arbeit ein Konzept zum Lernen unter Verwendung von Mobiltelefonen dar. Basierend auf aktuellen Forschungsergebnissen im Bereich des mobilen, sowie des Spiel-basierten Lernens, möchte ich hier einen Ansatz vorstellen, der die beiden Forschungsbereiche kombiniert. Es werden dabei Lernstrategien mit Spaßelementen verbunden, um vor allem die Zielgruppe von 18-24 jährigen StudentInnen anzusprechen. Die zugrunde liegenden Forschungsergebnisse, sowie die gesamte System-Architektur werden ausreichend dargestellt, das Hauptaugenmerk liegt jedoch auf der technischen Spezifizierung und Implementierung der mobilen Anwendung.

FIFTH THIRD EX1062
PART 2 of 3
Page 174 of 379

Abstract

Nowadays almost everybody owns a mobile phone and carries it around the whole day. It is obvious, though, to use it not only as a phone, but also as a tool to do other things, like reading news or watching videos while travelling from or to e.g. work or school. Cell phone manufacturers acted on that idea, by producing so called *Smart Phones*, that are equipped with e-mail clients, web-browsers and multimedia-players, beside other useful applications.

Going a step further, this work presents the concept of learning with someone's mobile- or smart-phone. Based on state of the art research in the fields of mobile learning and game-based learning, I am presenting the implementation of a mobile game-based learning approach. It incorporates learning strategies with fun elements to make it more attractive to the target group of students at the age of 18-24 years. While describing research basics, as well as the whole system-architecture, the work focuses on the technical specification and implementation of the mobile application.

FIFTH THIRD EX1062
PART 2 of 3
Page 175 of 379

Acknowledgements

My greatest debts are owed to both my family and my girlfriend, who have always and tirelessly supported me for my whole study, and beyond.

Furthermore, I want to thank all my colleagues of the Studio Smart Agent Technologies, SAT (which is part of the Research Studios Austria), as well as all other participants of the EU project *Mobile Game-Based Learning* (*mGBL*). Special thanks, however, go to Ulf Harr (SAT) who designed the authoring interface (appendix C) as well as the controls graphic (figure 5.5), Erik Fürst (SAT) for creating the content images (figure B.2) and Brigitte Krenn (SAT) for fruitful discussions. Moreover, I want to thank Florian Koch (Sensomatic) who created the design elements used in the client-application (e.g. the boss images), Marko Justinek (University of Maribor) for providing the basic version of figure 5.2, as well as Alice Mitchel (Inspire) who created the majority of the presented content of the learning games. The images of the mobile phones (figures D.1 to D.4) are taken from the manufacturers' websites. Parts of this master thesis were incorporated in *mGBL* deliverables and publications.

Special thank also goes to my carer Prof. Mag. Dr. Silvia Miksch, who supported me continuously with precise and helpful advice.

FIFTH THIRD EX1062
PART 2 of 3
Page 176 of 379

Contents

1 Introduction **1**
- 1.1 Problem . . . 1
- 1.2 Goals . . . 1
- 1.3 Structure of the Work . . . 2

2 Learning Theories **3**
- 2.1 Formal, Non-Formal and Informal Learning . . . 3
- 2.2 Game-based Learning . . . 4
  - 2.2.1 Game Definition . . . 4
  - 2.2.2 Games and Computer Games . . . 5
  - 2.2.3 Benefits of Game-based Learning . . . 5
  - 2.2.4 Selecting or Finding a Game . . . 6
  - 2.2.5 Characteristics in the Learning Setting . . . 6
  - 2.2.6 Introducing a digital game into a course . . . 7
  - 2.2.7 Debriefing . . . 7
  - 2.2.8 Beyond Education . . . 7
- 2.3 Mobile Learning . . . 7
  - 2.3.1 Benefits . . . 8
  - 2.3.2 Challenges . . . 8
- 2.4 Mobile Game-based Learning . . . 9

3 Mobile Applications **10**
- 3.1 Specialities of SW-Development . . . 10
  - 3.1.1 Limitations . . . 11
  - 3.1.2 The Software Development Process . . . 12
- 3.2 Technology . . . 13
  - 3.2.1 J2ME . . . 13
  - 3.2.2 Symbian OS . . . 25
  - 3.2.3 .Net Compact . . . 26
  - 3.2.4 Flash Lite . . . 29

4 Mobile Game-Based Learning **32**
- 4.1 Web-Platform . . . 32
- 4.2 Templates . . . 34
- 4.3 Distribution . . . 34
- 4.4 Communication . . . 34
- 4.5 Roles & Tasks . . . 34
  - 4.5.1 Roles . . . 35
  - 4.5.2 Tasks . . . 35

FIFTH THIRD EX1062
PART 2 of 3
Page 177 of 379

4.5.3 Server-side & Client-side Tasks . . . 35
4.6 Focus . . . 36

5 Template Specification **37**
5.1 Overview . . . 37
5.1.1 Components . . . 37
5.1.2 Game-Flow . . . 37
5.1.3 Game Levels . . . 40
5.1.4 Points . . . 40
5.1.5 Progression . . . 40
5.1.6 Game Saves . . . 40
5.1.7 The Boss . . . 40
5.2 Component 1 — Quiz . . . 40
5.2.1 Principle of game play . . . 40
5.2.2 Jokers . . . 42
5.2.3 Feedback . . . 42
5.2.4 Rewards/Penalties . . . 42
5.3 Component 2 — Simulation . . . 43
5.3.1 Principle of game play . . . 43
5.3.2 Self-Evaluation . . . 43
5.3.3 Feedback . . . 43
5.3.4 Jokers . . . 43
5.3.5 Rewards and Penalties . . . 43
5.3.6 Authoring Example . . . 45
5.4 Scoring . . . 48
5.4.1 Quiz . . . 48
5.4.2 Simulation . . . 49
5.4.3 Rewards & Penalties . . . 49
5.4.4 Hints . . . 49
5.4.5 Time-Aspect . . . 49
5.4.6 Authoring . . . 49
5.5 Non-Features . . . 49
5.6 Controls . . . 50

6 Template Implementation **52**
6.1 Standards . . . 52
6.1.1 API . . . 52
6.1.2 Devices . . . 52
6.1.3 Screen Sizes . . . 52
6.1.4 Multimedia Content . . . 53
6.2 Architecture Overview . . . 53
6.3 Package Structure . . . 55
6.4 Software Design . . . 56
6.4.1 Controller . . . 56
6.4.2 Threading . . . 58
6.4.3 XML-processing . . . 58
6.4.4 UI-Framework . . . 58

FIFTH THIRD EX1062
PART 2 of 3
Page 178 of 379

6.4.5 Representations . . . 58
6.4.6 Connection to the world . . . 59
6.4.7 User Profiles . . . 59
6.4.8 Utils . . . 59
6.5 Datamodel . . . 59
6.5.1 Game Information . . . 61
6.5.2 General Information . . . 61
6.5.3 Quiz . . . 61
6.5.4 Simulation (SimCeLO) . . . 62
6.5.5 Feedback . . . 63
6.5.6 Self-Assessment . . . 63
6.5.7 Information (InfoCeLO) . . . 63
6.5.8 System generated Information . . . 64
6.6 Settings, Configuration . . . 64
6.6.1 Player . . . 64
6.6.2 Author . . . 64
6.6.3 Developer . . . 65
6.7 Server Communication . . . 65
6.7.1 Distribution . . . 65
6.7.2 Game Start . . . 65
6.7.3 Game end . . . 65
6.7.4 Feedback . . . 67
6.7.5 Component-Synchronization . . . 67
6.8 File Structure . . . 67
6.9 Highscores . . . 67
6.10 User Profiles . . . 68
6.11 Game Deployment . . . 68

7 Evaluation **69**
7.1 Hype or Tripe . . . 69
7.2 Feedback from User Trials . . . 69
7.3 Ideas for Improvements . . . 71

8 Conclusion **73**

9 Bibliography **74**

A File Format **77**

B Game Examples **81**

C Authoring Interface **84**

D Test-Devices **88**

FIFTH THIRD EX1062
PART 2 of 3
Page 179 of 379

List of Tables

3.1 The most important *JavaME* Libraries 16

4.1 Different user-roles. 35
4.2 Tasks of the most important roles. 35
4.3 Tasks on the server and the client 36

5.1 Simulation authoring example 46
5.2 Non-Features 50

6.1 The application's package structure 55

FIFTH THIRD EX1062
PART 2 of 3
Page 180 of 379

List of Figures

3.1 The Java family at a glance . . . 13
3.2 The relation between J2SE, CDC and CLDC . . . 14
3.3 *JavaME* software layer stack. . . . 15
3.4 The MIDlet lifecycle. . . . 19
3.5 Overview of *JavaME*'s Low- and High-Level GUI. . . . 21
3.6 .Net's Common Language Infrastructure (CLI) . . . 28
3.7 Flash MX IDE . . . 30

4.1 Overview of the web platform . . . 33

5.1 Game flow overview . . . 38
5.2 Overview of the whole game . . . 39
5.3 Gameplay of the quiz component . . . 41
5.4 Gameplay of the simulation component . . . 44
5.5 Controls . . . 51

6.1 Overview of the whole system from a client's perspective . . . 53
6.2 The game creation process . . . 54
6.3 Overview of the whole system from a server's perspective . . . 54
6.4 *Ahead of the Game* design overview . . . 57
6.5 *Ahead of the Game* data model . . . 60
6.6 Client-server communication . . . 66

B.1 Exemplary game-flow of a quiz component. . . . 82
B.2 Two incidents of a simulation component . . . 83

C.1 Authoring card for a Single-Click Question. . . . 85
C.2 Authoring card for a Multiple-Click Question. . . . 86
C.3 Authoring interface for a simulation. . . . 87

D.1 Specification of Sony-Ericsson's m600i . . . 89
D.2 Specification of Sony-Ericsson's v630i . . . 89
D.3 Specification of Nokia's 6230i . . . 90
D.4 Specification of Sony-Ericsson's P990i . . . 90

FIFTH THIRD EX1062
PART 2 of 3
Page 181 of 379

Chapter 1

Introduction

1.1 Problem

Nowadays information is very important for nearly everybody. Take *Wikipedia*[1] as an example. It covers 1.47 million articles, and grows per approximately 1820 every day[2], while the library is consulted more than 6 Million times a day[3].

On the one hand, we simply don't know how to keep up to date with masses of information, that may come from newspapers, books, scientific papers or *just* the Internet. And we all know the problem of not using already gathered wisdom regularly. That is, what makes us lose it piece by piece.

On the other hand, knowledge builds the basis for decision making. But knowledge is not the only factor – the other is practice. Especially in life-threatening areas (e.g. hospitals' emergency rooms), it is essential to make a decision in a short amount of time.

Therefore, it is necessary to create a space, where people can practice, and get feedback about the quality of their choices, but without the fatal consequences of the daily life.

1.2 Goals

The overall goal is to improve the effectiveness and efficiency of learning, as well as decision making, in a cognitive and emotional way. To accomplish this overall goal, it can further be divided into the following sub-goals:

- Develop (innovative) learning models that can be applied to different learning domains.
- Specify and implement, at least prototypical, games, based on the above mentioned learning models.
- Examine those learning models in user tests to derive their effectiveness and efficiency.

The EU project *Mobile Game-Based Learning* (*mGBL*)[4] has been originated to address exactly the above defined goals by developing educational game templates for mobile phones. A server side tool is used by teachers/game authors to create several game instances by combining a game template with different contents.

The following list sketches the basic steps defined by *mGBL*:

[1] `http://en.wikipedia.org`, accessed at 10.10.2007
[2] New English articles in October 2006 [Zacb].
[3] Accessed English articles in October 2004, which is the last recorded date in [Zaca].
[4] For more details concerning the project see chapter 4

FIFTH THIRD EX1062
PART 2 of 3
Page 182 of 379

1. Plan and carry out a user-requirements analysis for revealing the needs of all stakeholders (teachers, students, IT-staff, etc.).
2. Consider and choose, or create pedagogical frameworks and learning models for supporting students in developing learning and decision making skills.
3. Design and implement a prototype server platform and client software.
4. Plan and carry out user-trials of the implemented software, in a way that allows to verify the effectiveness of the developed learning models.

The above listed goals apply to this work, too, although differently weighted. For example, *mGBL* demands for implementing a server platform and at least three game templates, while this work focuses on the one game template, it is based on. For more details about the difference of the *mGBL* project and this work, please see section 4.6. However, the next chapter explains the organisation of this work, and makes clear how the chapters are weighted.

1.3 Structure of the Work

Starting with state of the art research in the field of learning, the first chapter (chapter 2) explains the benefits, and therefore reasons for choosing mobile learning as well as game-based learning. The following chapter (chapter 3) lists the most recent development technologies, their benefits and short comes, and goes into more detail with the technology, chosen for this project (*JavaME*). The chapters 5 and 6 build the main part of this work, because they describe the specification and implementation details of *mGBL*'s game template 1 *Ahead of the Game*. You get insights about the two components of the template, quiz and simulation, how they could be combined, and their interaction with the outside world. A detailed insight about the implementation, gives a better understanding of how the application can be used, or extended with new ideas. At the end of this work I present findings of the conducted user trials, as well as ideas for further improvements (chapter 7).

FIFTH THIRD EX1062
PART 2 of 3
Page 183 of 379

Chapter 2

Learning Theories

We have moved from "Learning for life", which is limited to the period when we were pupils or students, to "Lifelong Learning", a process accompanying us for the rest of our lives.

We also have moved from teaching to learning and from authoritarian to authority in education. Formal education plays a limited part in our life, and we all know that people learn a lot from TV, newspapers, the Internet and other sources [PKD04]. We are now at a point where the society accepts (even slowly) that game-based approaches or additions in education can lead to good results — or even better ones than traditional approaches. We also know, that we are living in a fast moving world where time is limited, and we have to make the best of our opportunities and use the time while travelling to work or school for useful tasks like learning. That's why *game-based learning* and *mobile learning* as well as the combination of both areas (*mobile game-based learning*) have a very high potential to expand, and thus improve, our well known educational systems.

2.1 Formal, Non-Formal and Informal Learning

The government needs formal assessment of individuals, where companies and organisations are interested in both formal and non-formal education for their employees to denounce their new skills and knowledge. But, what happens behind the scenes, by cooperating with other people, is informal learning (please find exact definitions below).

Sweden, for example, realized that non-formal and informal learning is very important for the development of individuals, and is therefore accredited by the government [PKD04].

Formal Learning takes place in an education or training institution and is well structured which means that the objectives and the time frame for learning are clearly defined and there is some kind of learning support – a teacher or guide – available. It is intentional from the learner's perspective. Formal learning requires measurable knowledge and skills and is therefore demanded by state, because otherwise assessment seems not possible.

Non-formal Learning takes place aside the mainstream system of education and training and does typically not lead to formal certificates. It can be provided in the workplace, through activities or civil organisations or political parties. Like formal learning, it is also intentional from the learner's perspective. Non-formal learning does not verify skills and knowledge, but rather makes it visible to the learner selves and others. It is characterized by how, not which, gathered skills and knowledge are used.

FIFTH THIRD EX1062
PART 2 of 3
Page 184 of 379

Informal Learning is a natural aspect of everyday's life. Unlike others, informal learning is not necessarily intentional from the learner's perspective. It, therefore, may not be recognized by individuals themselves. Because it is most often unintentional and not recognized, informal learning is neither structured, nor does it lead to certification.

2.2 Game-based Learning

> "Education is what remains, when what was learned has been forgotten." [PKD04]

Therefore, education should be the active engagement of learners with the subject matter, instead of its passive reception.

Learning through games is a concept that has always existed in the human history. Children playing with each other and young adults playing computer games have many in common — motivation is high, and they can play for hours even not realizing that time passes by. That is exactly what teachers wish in their own education.

Pivec et al. [PKD04] describe an example of a medical student, who is "playing being a doctor", where in the early stages of education – maybe even before the education starts – this play has indeed a large element of fantasy in it. But this fantasy should not be viewed negatively, e.g. as self-delusion, but rather positively, as high aspiration. During his education the medical student is increasingly asked to play the role of a medical practitioner, where in this role his skills and knowledge become more and more cultivated. The role-play is thus used not only as a teaching method in medical education, but also as a method of assessment. The process by which the behaviour of the student ends to be a role-play and starts being reality is a gradual and often invisible one. Safety is (beside trust) an essential element of the social contract of a game. It is a prerequisite for a player to be comfortable enough to dunk into the game and it's social context [SZ03]. Obviously, there is nothing trivial or frivolous about these activities that are orchestrated by medical educators, nor should it be in other areas of education.

2.2.1 Game Definition

A game is...

> "A contest with rules to determine a winner." [Wor]

> "A pursuit or activity with rules performed either alone or with others, for the purpose of entertainment." [Wikk]

> "Reduced to its formal essence, a game is an activity among two or more independent decision-makers seeking to achieve their objectives in some limiting context. A more conventional definition would say that a game is a context with rules among adversaries trying to win objectives." [Abt87], cited in [SZ03]

> "Any overt instructional or learning format that involves competition and is rule guided." [DRL96], cited in [PKD04]

FIFTH THIRD EX1062
PART 2 of 3
Page 185 of 379

Summing up these definitions, a game consists of rules, activities, players or decision-makers, a winner or winners, and is set up in a context where a competition takes place. There may be more elements to be found in other definitions, which may raise the complexity of a game or the understanding of games. However, those elements must be considered when defining a new game (please find further information on defining a game in [SZ03]).

2.2.2 Games and Computer Games

Research on the potential of computer games in educational contexts dates from the earliest days of the microcomputer ([Mal81a, Mal81b], cited in [PKD04]).

The key advantage of computer games over conventional games is their knowledge about the status of the player and the game itself. Motivation is generated by solving challenging but achievable tasks — beside active participation, specific and prompt feedback and some kind of uncertainty. Based on the current game status, the program is able to continuously motivate players by analysing their competence-level and adopt the difficulty accordingly. This way, the software can start with a difficulty level the player is comfortable with, and increase it progressively to set it always a bit beyond the player's comfort level. Thus, by providing maximum excitement, the player is always moving forward at an optimal learning rate. [PKD04]

2.2.3 Benefits of Game-based Learning

The most important benefits of game-based learning are the learner's motivation and engagement, here motivation is considered as one of the key factors contributing to effective learning. Having fun is an additional factor that is central for effective learning. [PKD04]

To achieve the learner's motivation, games do not have to be novel, or challenge them (apparently) with the usage of complicated or sophisticated equipment. Such games rather provide players with continuous emotional conditions or psychological stimuli. These conditions ideally vary during the game play and might include satisfaction, desire, anger, interest, excitement, enjoyment, pride of achievement, peer recognition, and many more.

Compared to traditional learning formats, video and computer games create a unique form of "play environments" where motivation and fun provide a powerful learning experience. The player, irrespective of age, is acting in an active and performance-based play environment, including all related activities such as active discovery, analysis, interpretation and problem-solving, sometimes supplemented by physical activities.

Game-based learning approaches, help people to develop skills, that are necessary to transfer information and skills from one situational context to another. To achieve this, coherence between the game and real life contexts is extremely important.

Security is another great advantage of games, and especially digital games, which enable learners to act in a risk free virtual environment without the sometimes fatal consequences of real life. Players have the opportunity to make (wrong) decisions, evaluate the consequences, and refine their judgement and decision making process.

Finally it should be said that it may be more effective to use selected elements of a game that seem appropriate, rather than the game as a whole, in a curriculum. [PKD04]

FIFTH THIRD EX1062
PART 2 of 3
Page 186 of 379

2.2.4 Selecting or Finding a Game

The first task in finding an appropriate game to support learning is to identify the educational objectives that should be achieved.

Pivec et al. define three basic approaches for selecting a game. The *mGBL* project adds another one:

- Use an existing – commercial or non-commercial – game and embed it in the education process. Games like *SimCity*[1] may be used for example in the context of ecology or resource management. In contrast, simulation games like *The Sims*[2] or *Second Life*[3] may rather be used for social and behavioural topics. Several other commercial games may be used for learning areas like history, maths, logistics, and others.

- Use an existing game that has been specifically developed for an educational purpose. The most educational games can be found in the areas of business studies and political science. [ASS71, Pre01] cited in [PKD04]

- Build a new game for the desired purpose. As you might know there exist some freely available, as well as commercial, tools that help developing a computer game. As an example think of the *Unreal Engine* that originally has been created for the Ego-Shooter *Unreal* and was used in many other games so far [Wikj]. While commercial games cost about 1 million USD to 50 million USD to bring to the market place where the costs of 350.000 USD for the Unreal Engine are legitimate, there is a much lesser budget for educational contexts, what makes such engines less attractive [Wikj].

- Adapt a game template, or some other kind of skeleton, that can easily be modified to the author's needs. This seems to be the most attractive one of the creational approaches, because it combines a manageable implementation effort at limited costs with the potential of individual and creative fabrication of professional games.

2.2.5 Characteristics in the Learning Setting

Some key characteristics of the course should be taken into account for the integration of a game-based approach. Some of the most important characteristics are [PKD04]:

- the size of the students' group — while some activities are only possible with a certain number of students, a large group size may, in certain situations, create logistical problems.

- the information and communication technology skills of the students group

- the technical resources which are available for the student group to use — hard and software must be accessible to everyone who wants, or needs, to use it.

- the technical and educational support, the teacher can count on

[1] `http://simcity.ea.com/index.php?`, accessed at 10.10.2007
[2] `http://thesims.ea.com/`, accessed at 10.10.2007
[3] `http://secondlife.com/`, accessed at 10.10.2007

FIFTH THIRD EX1062
PART 2 of 3
Page 187 of 379

2.2.6 Introducing a digital game into a course

Game-based approaches will usually be included in a course as one of several elements. Its aim will differ from course to course. It may be one of the following [PKD04]:

- Usage as introduction to a certain topic or activity to trigger further and deeper engagement with the topic. Students often want to know facts behind their favourite computer games, what makes it ideal to introduce, for example, a History lesson with the game *Age of Empires*[4].
- Increase – at least the likelihood of – student participation in formative assessment. The motivation of awarding a grade is often not high in classes, but can be increased by challenges through competition or time pressure. The competition might be between individuals, teams, or by competing with the student's own "best performance".
- A game-based approach may create a simulation environment where students have a safe place to practice in a harmless way. The benefits of such simulations can often not be provided otherwise, what makes it ideal for professional areas like military, medicine, nursing, surgery, teaching and engineering.

2.2.7 Debriefing

To provide the maximal educational benefit, an explicit debriefing may be necessary for game-based learning approaches. [PKD04]

2.2.8 Beyond Education

Like computer technology holds a fascination for some people, encouraging them to engage with the technology behind the scene, starting programming, we can hope that a similar fascination happens to students who are not that technological affine, digging into a topic presented in a game-based approach.

2.3 Mobile Learning

> "Mobile learning is unique in that it allows truly anywhere, anytime, personalised learning." [Att04]

Mobile-Learning (M-Learning) is a new approach to combine the mostly static *Electronic-Learning* (E-Learning) with the power of mobile information and communication technologies. Associated with this topic are challenges in the areas of didactics, technology and organization of the learning process, business models are, in particular, very important, to show non-enthusiasts the power of new learning approaches. Additionally we must keep in minds, that M-Learning is very different to E-Learning, and direct transportation of content from the latter to the first area won't work for the most cases [SHB04]. Leon Rodin therefore started a research with the aim of converting E-Learning to M-Learning content, which concluded that a complete re-work of the content is necessary, otherwise it won't work [Rod04].

[4] `http://www.microsoft.com/games/pc/age3.aspx`, accessed at 10.10.2007

FIFTH THIRD EX1062
PART 2 of 3
Page 188 of 379

2.3.1 Benefits

Jill Attewell [Att04] identifies some key benefits that come from mobile learning, which have been observed during the *M-Learning*[5] project.

Mobile learning helped learners to improve their literacy and numeracy skills. That mainly comes from the fact that teachers recognized already existing abilities for the first time. One mentor even reported that a pupil improved reading skills just because of switching the media. Because of negative educational experiences he refused reading books, but he indeed read texts and information regarding the device very well.

Mobile learning can be used to encourage both independent and collaborative learning. Some people are glad about the opportunity to work at home at night, when they don't feel under pressure or under control. Others like working together in teams, because it incites to achieve a common goal, or something useful for the group. M-Learning can encourage both, depending on how it is used.

Mobile learning helps learners to remain more focused for longer periods. A mentor reported that a group of pupils was focused up to two hours, where, normally, it was difficult to keep them focused for 15 minutes. Attewell admits that this could be a result of the novelty of using mobile phones for educational purposes. Whether this is true or not will become clearer over time.

Mobile learning helps to raise self-esteem and self-confidence. Both are not related to the learning material, but to the learners' general attitude. While self-esteem has been reported to come from the teachers' trust giving expensive and sophisticated equipment to pupils, the raised self-confidence seems to be an effect of the trial-and-error principle of most electronic devices.

2.3.2 Challenges

Wikipedia identifies challenges in the field of mobile learning that are divided into two categories [Wikg]. Based on these, I enhanced the two lists with experiences due to my work with mobile phones, and findings of research papers according to mobile learning.

Technical challenges include:

- Connectivity — Wap, GPRS, Bluetooth, WLAN, etc. can be used for interacting with other devices. But not all phones are equipped with all these technologies, what makes it essential to support a few of them and making it possible to select one's preferred technology.
- Battery life — stand-by times of modern devices are up to one or two weeks, though this shortens to a few hours when the device is in use. Especially when the mobile phone is used for purposes it was intended for, this could let battery power of a few minutes for other activities like learning. Therefore it is essential that mobile learning

[5] `http://www.m-learning.org`, last accesses on 10.10.2007

FIFTH THIRD EX1062
PART 2 of 3
Page 189 of 379

applications support small lecture chunks, that can easily and automatically be saved in short intervals.

- Interacting with small devices — it's a challenging task to design a user interface for mobile phones (which usually lack of the input devices available for computers) that suits to many different devices allowing to control them in an easy and intuitive way.
- Displaying useful content on small-screen devices — content has to be adopted to the small screens with low resolutions that are typical for mobile devices (in average such screens are about 5cm large and have a resolution of 176x208 to 240x320 pixels).
- Presenting different kinds of media — due to the vast variety of used technology, and the (often) low processing power of current mobile phones, it is often hard to find a common ground for multi-media content (video- and audio-snippets, office files like Word, Excel, PDF).

Social and educational challenges include:

- Assessment of learning outside the classroom — we know from [Att04] that mobile learning helps to recognise existing abilities, and assessment should accommodate that, even though as just one facet in the assessment process.
- Tracking of results and proper use of this information — the results can be used in different ways, like for assessment, or for constructive work that banks on these outcomes.
- Support of learning across many contexts — especially with reduced budget that doesn't make it possible to develop discrete mobile applications for each scope.
- Developing an appropriate theory of learning for the mobile age — this learning model ideally makes use of the potential of different available technologies (digital cameras, GPS-Navigation, etc.), by combining them into one learning scenario. This makes the education approach more attractive for younger learners, and introduces variety for people of every age to keep them motivated with the topic.
- Develop and preserve ethical principles like confidentiality of data, anonymity of participants or a *Netiquette*, similar to those we are already familiar with, from e-mail or chat. [TB04]

2.4 Mobile Game-based Learning

Mobile game-based learning is the effective combination of M-Learning with elements of game-based learning to produce appealing and motivating learning software.

Graham Brown-Martin describes an opportunity we now have at schools:

> "Actually, we don't have a choice. Young learners already have this technology [mobile phones], and for the very first time schools don't have to provide that. They just have to embrace it in a meaningful and useful way." [Mat]

This opportunity can easily be generalized to many other formal, non-formal and informal learning environments, and offers great potential in these educational areas.

FIFTH THIRD EX1062
PART 2 of 3
Page 190 of 379

Chapter 3

Mobile Applications

Some of the most important technologies for developing applications or games for mobile devices are: *J2ME*[1], *Flash Lite*[2], *.Net Compact*[3] and *C++* (for Symbian OS)[4]. The former two are independent of the operating system, but still require some kind of interpreter on the host system, the latter two are built upon the APIs of specific operating systems, and can therefore only be executed in these environments. Java is the most common and most supported programming language for mobile devices, where C++ became more popular in the last few years, because Symbian OS gained a larger market segment as operating system for cell- and smart-phones. Dot-Net Compact, which can only be used on Windows CE powered devices, and Flash Lite are the least popular of the here presented technologies.

All technologies have some common ground, that is, they have to cope with reduced processing power, memory, and small, differently sized displays. But each has its advantages, like better performance (C++), easy graphics support (Flash Lite) or a wide distribution (*JavaME*). However, the used technology depends, like it does for developing desktop applications, on the specific use cases.

This chapter presents some characteristics that should be kept in mind when developing software for mobile devices (especially mobile phones), and concludes with an overview of the above listed technologies. It focuses on *JavaME*, because it is the technology used to implement *mGBL*'s game template 1 *Ahead of the Game*.

3.1 Specialities of SW-Development

When you start working with *JavaME*, you might think that there is no difference to developing desktop or server applications using Java. This could be a great mistake.

A blue-eyed *JavaME* development project may proceed like this:

1. You design the software in the object-oriented manner, you are used to.
2. Implementation starts, and you find out that you have to be very creative in replacing expected API functionality, that is not available for *JavaME*.
3. Bugging around with creating the desired GUI.

[1] `http://java.sun.com/javame/index.jsp`, accessed at 10.10.2007
[2] `http://www.adobe.com/products/flashlite/`, accessed at 10.10.2007
[3] `http://www.microsoft.com/germany/msdn/library/mobility/windowsmobile/pocketpc/DasNETCompactFramework.mspx`, accessed at 10.10.2007
[4] `http://www.symbian.com/`, accessed at 10.10.2007

FIFTH THIRD EX1062
PART 2 of 3
Page 191 of 379

4. It is necessary to shrink used resources (e.g. images) to undergo the file size limits of the mobile device that's used for early tests.

5. Early tests show that the application reacts very slowly on user input, what leads you to get familiar with emulator-based debugging and on-device-debugging.

6. Mastered points 1 to 5, you want to show the application to a friend and transfer it on his/her device, where:

   - it does not work at all
   - it looks completely different
   - the application crashes arbitrarily

7. At last you end up in adopting the application to various mobile phones which doubles your previous efforts.

Though, this scenario describes the worst case, it shows that it is very essential not to underestimate a "small Java application". Mostly you won't be able to finish the project; it would take you nearly the whole planned project time to implement only half of the above sketched tasks, which do not even include changes in the software itself, e.g. change requests, or changes in the software architecture.

To circumvent such disastrous projects, you may want to get familiar with some basics, that are special to mobile devices and their software development. Based on this knowledge, it is easier to estimate a software development project plan.

3.1.1 Limitations

Often people are not aware, that mobile phones are not as powerful as PCs, and expect that the same "software magic" is possible. Here are some important limitations, listing the more obvious ones first:

- The *screen size* is limited and different for many devices. MIDP specifies a minimum of 96x54(!) pixels [Sch07]. Displays of modern mobile phones have a resolution that lies between 128x128 and 240x320 pixels. That kind of variations is typical for mobile devices, and very unique compared to PC monitors and their resolutions, for example.

- Usually a cell phone has *a joystick* and *12 keys* (10 for the numbers 0 to 9 and 2 soft keys) to navigate through menus and dial numbers[5]. A computer keyboard has 104 or even more keys and is supported by a pointing device (usually a 3-button mouse).

- The *processor* is more powerful than the one used for travelling to the moon was, but that is still only about 1-15 percent of modern desktop CPUs.

- An obvious size is the *storage capacity* of a mobile phone, which often is expandable. But even more important is the available *RAM size*, which specifies the maximum transient memory capacity, and the heap-size, that further reduces the available RAM size for one instance of a (Java) Virtual Machine.

[5]Smart Phones are equipped with a full keyboard and a stylus for pointing tasks, but the interaction is still different combined to PCs, and you can't expect whether the user prefers the one input method (keys) or the other (touch-screen).

FIFTH THIRD EX1062
PART 2 of 3
Page 192 of 379

- Although, the *permanent memory* of modern mobile phones is large, some devices limit the size of .jar files (e.g. Nokia's 6230i accept only .jar files that are not larger than 512KB).
- *Access to the Internet* is often slow and expensive and the user needs to allow it (for exactly those reasons).
- Playback of *multimedia files*, like MP3 or video snippets, works quite well now, but is often limited to a few selected encoding-formats that differ from one to another device. Streaming is often not an option, because there is no standardized way to do that on mobile phones, why own algorithms have to be implemented, which leads often to bad results because of the devices' reduced processing power.
- Access to installed applications is often prohibited. It is not possible to (PC-like) drag&drop a number from the phone book to another application.
- The *API* of programming languages is limited due to the reduced RAM and CPU resources.
- Last but not least: *there are huge differences from one device to another.*

3.1.2 The Software Development Process

When you are developing software for mobile devices, you will experience that the software development process is quite different to desktop or server applications.

It starts with the *software design*, that needs to be less object-oriented to improve the performance [ZHKH07]. That means you have to find the correct balance between object-orientation (to benefit from readability, re-usability, etc.) and performance, on the target device. The more classes you have in your application, or the more instances you create, the slower it will get.

The *implementation* suffers from missing APIs, developers are used to from their work with desktop applications. This concerns not only high level GUI components, but also XML-Parser, or even logging APIs.

Source code compilation is not one task but consists of pre-processing (optional), compilation, obfuscation (optional), pre-verification, packaging, signing (optional) and jad-creation[6].

For *debugging* purposes emulators might help in the first instance, but can't replace a real device. And: because you don't have access to the underlying operating system, on-device-debugging is not an easy task. Some cell phone vendors and associated companies provide special development kits for on-device-debugging, unfortunately they don't work for all devices.

The *adaptation to different devices* may take up to 90% of the overall project time and costs, depending on the quantity of supported devices, the complexity of the GUI design, the used APIs and maybe some other factors. To reduce the efforts, or make them just less surprisingly, it may help to organize the adaption to different devices as a parallel task to the whole development process, like it is often done for quality assurance.

[6]These steps describe the *JavaME* build-process. The compilation process of other programming languages for mobile devices may be less complicated. However, knowing the process might help preventing troubles.

FIFTH THIRD EX1062
PART 2 of 3
Page 193 of 379

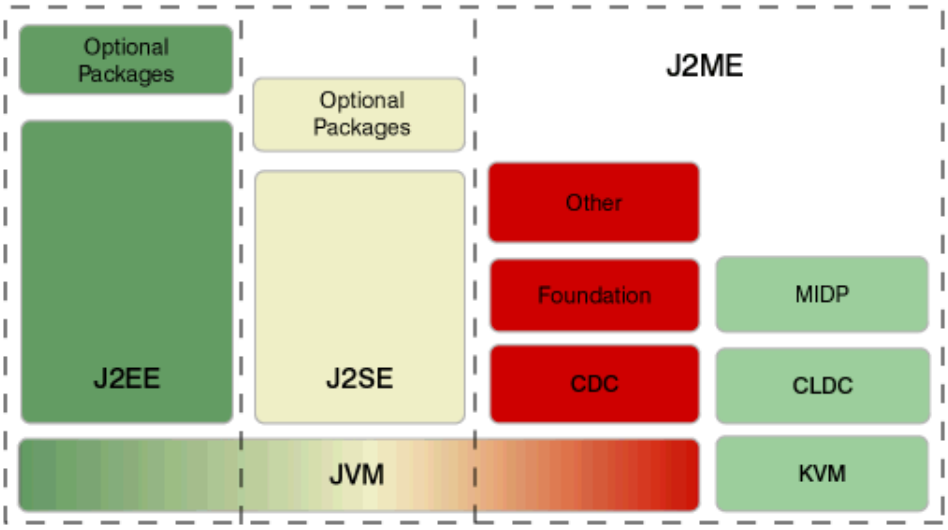


Figure 3.1: The Java family at a glance [Sun01]

3.2 Technology

3.2.1 J2ME

Java was invented in the year 1996 by Sun Microsystems and continually gained popularity, especially for desktop applications[7]. That's the reason, why Sun made no efforts to keep the resource demands low. When mobile devices become more popular, a subset of J2SE version 1.1 was used to build the basic J2ME API upon. Some classes where changed, some added[8], but most of them have been dropped. The result was a *Java Virtual Machine* (JVM) that uses only some kilobyte of memory, and therefore was called the *Kilobyte Virtual Machine* (KVM). Figure 3.1 shows an overview of the Java family including J2ME, J2SE, and J2EE. Additionally, figure 3.2 shows how different the mobile specifications (CLDC, CDC)[9] are from their desktop pendant.

Architecture

As for every device, the operating system is the interface to the device's hardware it is built upon. Based on the operating system, *JavaME* defines several layers that are arranged one above another (figure 3.3, [Sch07]):

- a configuration
- profiles

[7] The current version for desktop applications is 1.7 (specification started in August 2006, planned release is 2008) [Wikd].

[8] For additional classes the standard package `java.*` can't be used, because it contradicts the basic Java specification. For those classes the root package `javax.microedition.*` is used.

[9] The *Connected, Limited Device Configuration* (CLDC) and *Connected Device Configuration* (CDC) are described in more detail on the following page

FIFTH THIRD EX1062
PART 2 of 3
Page 194 of 379

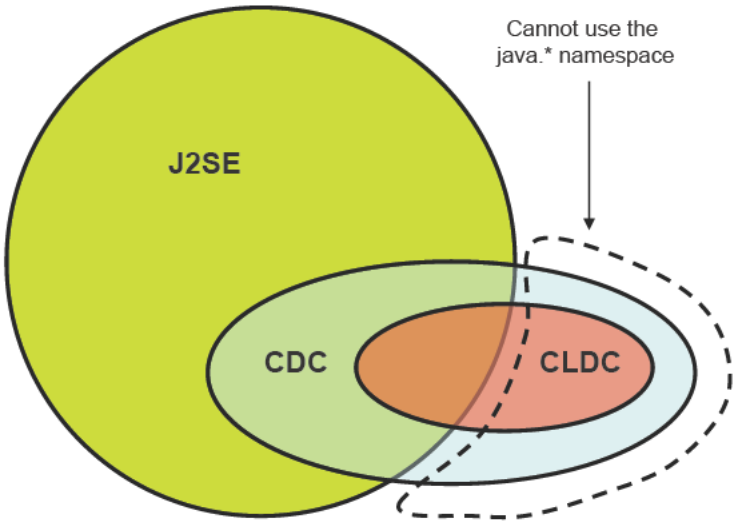


Figure 3.2: A schematic presentation of the relation between J2SE, CDC and CLDC

- optional packages

Configuration

The configuration defines minimal requirements to the host platform for a set of similar devices with identical or similar resources like screen size, CPU power, or the ability for floating point calculations – i.e. a horizontal class of devices. It further specifies the supported language scope (e.g. Generics[10] are not supported), the JVM's supported functional scope (e.g. the on-the-fly verification[11] is not supported; find further information on page 17), and the minimal set of library classes. Although the term *Java* in *Java Micro Edition* suggests that someone can use the full power of Java on mobile devices, only a subset of the language is supported.

Currently two configurations are in use[12]:

- The *Connected, Limited Device Configuration* (CLDC) is used for small mobile devices like cell phones.
- The *Connected Device Configuration* (CDC) is best known to be used for more powerful mobile devices like PDA's, but is intended to be used also for e.g. Set-Top-Boxes.

Version 1.1 of the CLDC specification, which was finally released in the year 2003, brought some benefits in contrast to its predecessor CLDC v1.0, for example:

- Floating point support; previously only integer values were supported

[10] Find a definition here: `http://en.wikipedia.org/wiki/Generics_in_Java`, accessed at 10.10.2007

[11] The Java verification process is described here: `http://en.wikipedia.org/wiki/Java_Virtual_Machine#Bytecode_verifier`, accessed at 10.10.2007

[12] Nowadays so called Smart-Phones are very popular, which combine the functionalities of a cellular phone and a PDA and therefore support both configurations.

FIFTH THIRD EX1062
PART 2 of 3
Page 195 of 379

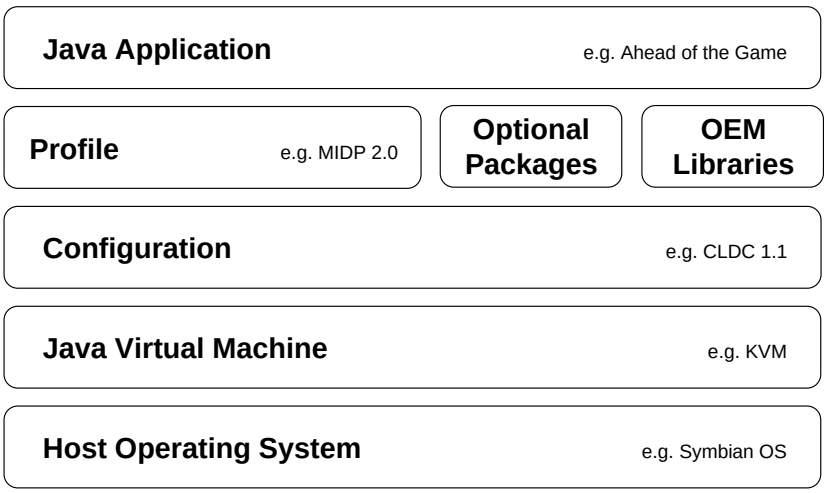


Figure 3.3: *JavaME* software layer stack.

- Weak reference support; a weak reference allows the garbage collector to remove a referenced object to save memory, where it can easily be restored, when needed.
- Time relevant classes (namely: `Calendar`, `Date`, `TimeZone`) have been redesigned to be more compliant to the J2SE standard.
- Threads became more compliant to the J2SE specification (the name of a thread has been introduced, as well as some new constructors).
- The minimum memory budget has been raised from 160 to 192 Kilobytes, mainly because of the introduction of floating point arithmetic.

Profile

A profile extends the configuration with more specific functionality like file-storage or visual components — i.e. a range of functions of a vertical market segment. In contrast to a configuration, a profile may specify optional units (that must not be implemented by vendors). The most important profile is the *Mobile Information Device Profile* (MIDP), which was initially released in September 2000, and has been enhanced to version 2.0 in November 2002.

Optional Packages

Optional packages are defined in the *Java Community Process* (JCP), together with all relevant vendors, although they are not liable for implementing them. Using optional packages enables software engineers to write code against defined interfaces of a horizontal class of devices, reducing configuration and/or implementation effort. Some popular, optional packages are shown in table 3.1, together with the above described configurations and profiles, to draw a complete picture.

FIFTH THIRD EX1062
PART 2 of 3
Page 196 of 379

Table 3.1: Overview of the most important configurations, profiles, and some selected optional packages for *JavaME* [Com].

JSR[13]	Package	Description
30, 139	CLDC, versions 1.0 and 1.1	Defines a standard platform configuration of the *Java™2 platform, Micro Edition* (J2ME™) for small, resource-limited, connected devices.
37, 118	MIDP, versions 1.0 and 2.0	Defines a profile that extends and enhances the *J2ME™ Connected, Limited Device Configuration*, enabling application development for mobile information appliances and voice communication devices.
82	Java APIs for Bluetooth	Bluetooth is an important emerging standard for wireless integration of small devices. The specification standardizes a set of Java APIs to allow Java-enabled devices to integrate into a Bluetooth environment.
120, 205	Wireless Messaging API (WMA), versions 1.1 and 2.0	Defines a set of optional APIs which provides standard access to wireless communication resources (like SMS or HTTP), designed to run on J2ME configurations and to enhance J2ME profiles with unique functionality.
135	Mobile Media API (MMAPI)	Specifies a small-footprint multimedia API for J2ME™, allowing simple, easy access and control of basic audio and multimedia resources while also addressing scalability and support of more sophisticated features.
179	Location API	Enables developers to write mobile location-based applications for resource-limited devices. (The API works on the J2ME CLDC v1.1 and CDC configurations.)
226	Scalable 2D Vector Graphics API	Defines an API for rendering scalable 2D vector graphics, including image files in W3C Scalable Vector Graphics (SVG) format.

OEM-Specific Libraries

Often a vendor wants to lay more power in the hands of developers, by letting them access functionality via the programming language, that is already implemented with his/her devices. Otherwise, it may be that there is no JSR specified or completed for a particular functionality, and therefore no standardized programmable interface is publicly available, yet. In these cases

[13] The *Java Specification Request* (JSR) is an integral element of the *Java Community Process* (JCP), which was founded to build a platform for extensions of the Java technology. A JSR is a technical proposal for extensions, where the final version of a JSR requires a reference implementation. [Wikc]

FIFTH THIRD EX1062
PART 2 of 3
Page 197 of 379

a vendor may define his/her own libraries for a hand of specific devices, or – more often – a certain product line (e.g. Nokia's Series 60). It is obvious that those libraries only work for the specified target devices. A good example is *Nokia's UI API*[14], which extends the MIDP 1.0 profile with transparency effects, polygons (e.g. triangles), image processing capabilities (e.g. rotating), as well as support for vibration and backlight controls.

Preverification

For security reasons, the Java Standard Edition verifies the bytecode of each class that is to be executed, when the application is about to be started [Wike]. This ensures, for example, memory protection and avoids the execution of invalid instructions. Usually this process requires a fair amount of heap-size and CPU power (and therefore time).

In the past this was no option for small devices like mobile phones, and is still deprecated. Therefore a pre-verification procedure has been developed that transfers most of the time and memory consuming procedures to the machine where the source-code compilation is done, thus a PC.

Pre-Verification is a two-fold procedure:

1. The pre-verifier processes a given *.class* file in the way, so that it remains semantically the same, but can be verified easily on the target machine.
2. The runtime-verifier does the remaining checks, before the *.class* file is interpreted by the virtual machine on the target device.

MIDlet and MIDlet-Suite

> "A MIDlet requires a device that implements Java ME, MIDP to run. Like other Java programs, MIDlets have a "compile once, run anywhere" potential. [...] MIDlet distributions also consist of a .jad file describing the contents of the .jar file." [Wikf]

A MIDlet has to fulfil the following requirements in order to run on a mobile phone:

- The main class needs to be a subclass of javax.microedition.midlet.MIDlet
- The MIDlet needs to be packed inside a .jar file (e.g. by using Sun's jar-tool)
- The .jar file needs to be pre-verified.
- In some cases, the .jar file needs to be signed by the mobile phone's carrier.

A MIDlet, therefore, can be considered as an application.

A MIDlet-Suite bundles one or more MIDlets into one .jar file, which is then distributed.

A .jar file is basically a .zip file with another file-extension, that incorporates the application's classes, a descriptive Manifest file, resources (e.g. graphics) and the used libraries (e.g. a XML-Parser library) [Wikb].

[14] `http://www.forum.nokia.com/info/sw.nokia.com/id/1f75c9a5-3aeb-42e9-890d-f06e3b2e94d9/Nokia_UI_API_Programmers_Guide_v1_1_en.pdf.html`, accessed at 10.10.2007

FIFTH THIRD EX1062
PART 2 of 3
Page 198 of 379

The .jad file, that always comes along with a .jar, is basically a copy of the Manifest, with some additional fields, that are primarily used for *Over The Air Provisioning* (OTA-Provisioning[15]).

Two fields are essential for OTA-Provisioning:

- *MIDlet-Jar-URL* specifies the location where the MIDlet-Suite (the .jar file) can be found on the server.
- *MIDlet-Jar-Size* declares the size of the referenced MIDlet-Suite in bytes. The value is used to:
  1. roughly verify the download candidate (by comparing the specified with the real file size).
  2. check whether the device has enough free memory for storing the application and if it supports applications with the given file size[16].
  3. ask the user whether s/he is willing to pay for the data-transfer of the given size.

Certification

Since the release of MIDP 2.0 it is possible to add a certificate to a .jar file to gain more rights for the MIDlet-Suite, e.g. access the storage system (the same that is used to store images, MP3 files, etc.). To achieve this, it must be able to authenticate the author, and the author must be classified as trustworthy. This is done by using asymmetric, cryptographic algorithms, based on the standard X.509, to sign the whole MIDlet-Suite [Sch07]. It is not sufficient that the author just signs the suite, because the user (i.e. his/her device) does not know whether the signature is trustworthy. Therefore the public key has to be signed by a *Certificate Authority* (CA), that guaranties the originality. Every device stores the public keys of the most important CAs and, thus, can check whether the MIDlet is trustworthy.

Because the signing process has to be done for nearly every mobile device separately, which obviously can become very expensive, this process is often neglected by most authors/publishers.

Programming Basics

A MIDlet has to extend `javax.microedition.midlet` and implement some basic methods:

Listing 3.1: Example for implementing a MIDlet

```
public class TestMIDlet extends MIDlet {
    protected void startApp() throws MIDletStateChangeException {
        ...
    }

    protected void pauseApp() {
        ...
```

[15] Find details here: `http://en.wikipedia.org/wiki/Over-the-air_programming`, accessed at 10.10.2007

[16] On some devices applications of various sizes can be installed, as long as there is enough free memory available. Others are more strict and specify a maximum jar-file size, e.g. 512KB for Nokia's 6230i [JPb].

FIFTH THIRD EX1062
PART 2 of 3
Page 199 of 379

```
    }

    protected void destroyApp(boolean unconditional) throws
        MIDletStateChangeException {
        ...
    }
}
```

These methods are essential, because they define a basic MIDlet-lifecycle as shown in figure 3.4. The three methods are called by the underlying operating system, to force a transition to the states *Paused*, *Active* and *Destroyed*.

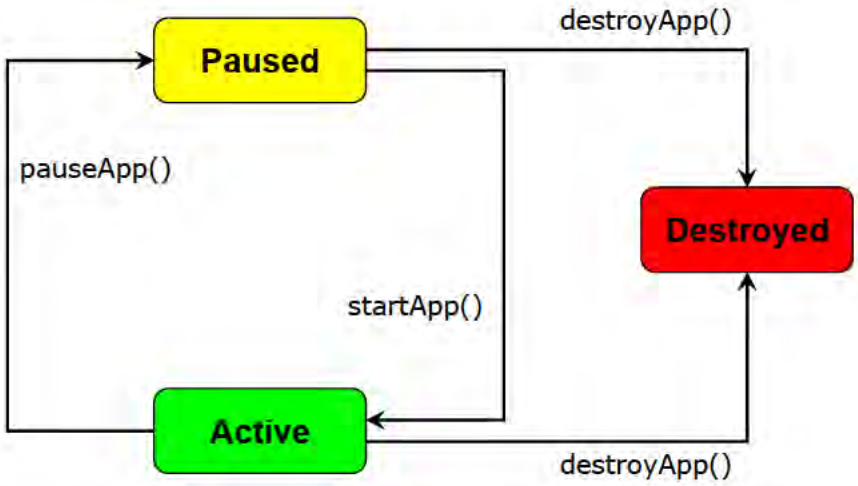


Figure 3.4: The MIDlet lifecycle.

- *Paused*: the MIDlet is initialized, but has no permission to output anything on the device's screen. Although background activities (e.g. calculations, network access) are not prohibited, they are not recommended, because another MIDlet may be active and competes for the occupied resources.
- *Active*: the MIDlet has access to the screen and receives user-input. Thus, this is the state, the application is usually executed in.
- *Destroyed*: the MIDlet stopped the normal program flow and released all resources. Depending on the application, the application's state may have been saved. Once reached, the state can not be changed any more and the program execution ends.

Access to Resources A *.jar* file may contain other resources than the used classes, which can be accessed by the MIDlet. Therefore, images, texts or other files may be included in the package and can be accessed by using the method `getResourceAsStream()` of `java.lang.Class`.

FIFTH THIRD EX1062
PART 2 of 3
Page 200 of 379

Because the classes of a package are already pre-verified to suppress manipulations, it is not possible to gain access to the *.class* files themselves (what makes *Reflection*[17] impossible).

Properties Two property types are accessible within a MIDlet (example code for accessing different properties can be found in listing 3.2):

- *System-Properties* define system specific attributes, like the user-defined language or available libraries (profile, configuration, or optional packages).
- *MIDlet-Properties* are configured in the *.jad* descriptor or the *Manifest* (find further information on page 17) and are either predefined MIDlet properties, or user-specified.

Listing 3.2: Example implementation that shows the access to different types of properties

```
public class TestMidlet extends MIDlet {
    ...
    private void getSomeProperties() {
        // System-Property (set by the host device)
        // e.g. ''MIDP-2.0''
        String midp = System.getProperty("microedition.profiles");

        // standardized MIDlet-Property
        // e.g. ''/img/icons/midlet.png''
        String icon = this.getAppProperty("MIDlet-Icon");

        // a custom property (specified in MANIFEST.MF, and the .jad
            descriptor)
        // e.g. ''192.168.0.1''
        String host = this.getAppProperty("Server-Address");
    }
    ...
    // implement the three abstract methods, beside other logic here
}
```

High- and Low-Level GUI The two approaches of programming a user interface lead to the differentiation of the corresponding APIs into a Low- and High-Level API or GUI, which is illustrated in figure 3.5.

Where the High-Level GUI is used to present the user a unique look on different devices, the Low-Level GUI can be seen as an empty sheet of paper (the `Canvas`) for drawing lines, curves, points or texts on. The High-Level API consists – similar to HTML – of elements like lists, text-boxes, alerts and forms (see figure 3.5), where the latter themselves are built upon elements like text fields, images or choice groups (similar to HTML's checkbox- or radiobutton-group). The concept of user input for primitive elements like a text-box or choice-group is obvious – via joystick and keypad – and transparent for developers. Because it is often necessary to react on user input, it is possible to register an `ItemStateListener`, which is similar to an *Observer* of the *Observer Pattern*. Additionally, each `Displayable`, thus all high-level elements and the low-level `Canvas`, can be assigned several `Commands` with associated actions. These can be used

[17] `http://en.wikipedia.org/wiki/Reflection_(computer_science)`, accessed at 15.11.2007

FIFTH THIRD EX1062
PART 2 of 3
Page 201 of 379

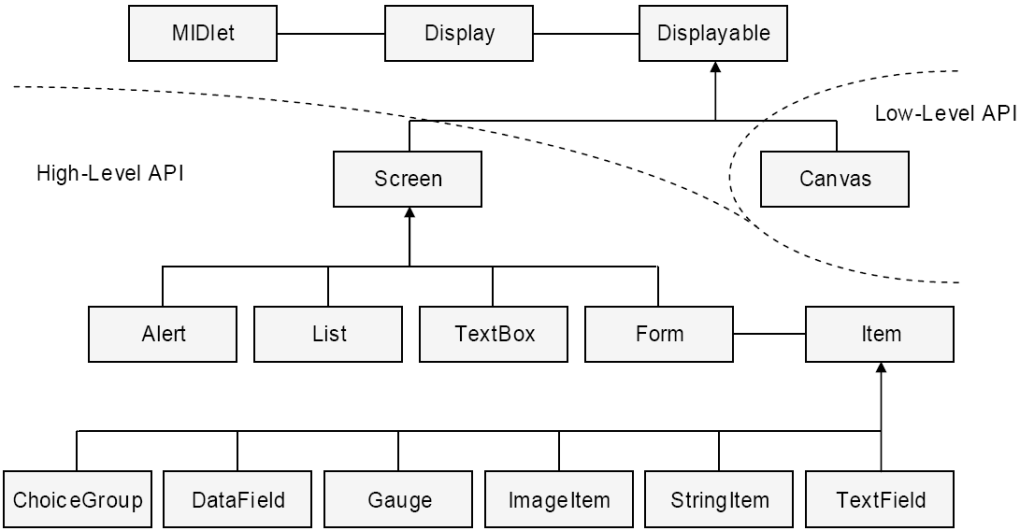


Figure 3.5: Overview of *JavaME*'s Low- and High-Level GUI.

to create menu entries, for general (e.g. "next" or "exit"), and specific (e.g. "add new item") commands.

The main usage of the Low-Level API can be found in mobile games, where developers need full control of the screen and the device's keys. Another reason for choosing the Low-Level API is when the user interface should look the same on every device. This is not possible using the High-Level GUI, because the design of high-level elements is not specified, and vendors tend to implement them differently e.g. to support corporate identity. For this special use case, the open source project *J2ME Polish* (find further information on page 24) has been founded, which lets developers use a special High-Level GUI, that can additionally be configured by using a subset of *Cascading Style Sheets* (CSS)[18]. As J2ME-Polish allows to specify a different CSS file for each device, the application can be customized to look the same on all handsets. The same way, this capability can also be used to define different look&feels for different devices or vendors.

Record Management System Because it is sometimes useful to store data on a physical, non-transient medium, MIDP specifies the *Record Store Management System* (RMS) that is accessible via a device-independent interface. It is a record oriented database with a unique name and fields of variable length and type. When a new record is added to the database, it is assigned a unique identifier, which has the same functionality as a primary key in relational databases. Records can be accessed directly (via its unique ID) or sequentially (one by one), where classes for sorting or filtering (that have to be implemented individually) may help to navigate. However, a RMS can not compete with relational databases like *MySQL* or *Oracle*.

All MIDlets of a MIDlet-Suite have the same access rights – read and write – to record stores that have been created by one if its MIDlets. Applications of other MIDlet-Suites are, per default, not able to access the RMS, although such access rights can be defined on creation.

[18] CSS has its roots in web-design, where it is used to specify a certain user-interface design (colours, text-sizes, ...).

FIFTH THIRD EX1062
PART 2 of 3
Page 202 of 379

IDEs

Integrated Development Environments (IDEs) are used for implementation efforts in various programming languages because they help writing error-less code in short time. They support developers in writing, compiling, testing, debugging and executing code and sometimes they transparently transfer the executables to a target host system.

In principle there is no difference between IDEs for developing desktop applications and those for developing *JavaME* applications. But, if you take a closer look, you will see that the whole software development process (as described in subsection 3.1.2) is quite different to those of desktop or server applications, and an IDE must support every phase of it.

Actually, there are two well established IDEs that support the development for *JavaME*:

- *Eclipse*[19] with the *EclipseME*[20] plug-in lets developers write, compile, obfuscate, pre-verify, package and emulate their applications. New versions of the plug-in allow the use of pre-processing instructions to customize the application based on certain criteria (e.g. the screen size).
- *Netbeans*[21] with the *Mobility Pack*[22] supports all the functionalities of EclipseME, but integrates it seamlessly into the IDE (both, the IDE and the plug-in are developed by Sun). Further advantages are for example the *Visual Design Editor*, for drag&drop GUI-creation or the integrated localization support.

To round up this overview, the list below shows tools that are IDE-independent, although they provide interfaces to be used in major IDEs like Eclipse or Netbeans:

- *Antenna*[23] specifies *Ant*[24] tasks for pre-processing, building, obfuscating, pre-verifying, packaging, signing and running MIDlets.
- *J2ME Polish*[25] is based on Antenna, but enhances the building process with the possibility of resource assembly based on vendors, devices or device characteristics.
- *Maven 2*[26] with the *j2me-maven-plugin*[27] (which is also based on Antenna) allows the integration within Maven2 project hierarchies and the use of other Maven2 plug-ins (e.g. the *maven-assembly-plugin*[28]).

Emulators/Emulation

Sun provides with the *Wireless Tool Kit* (WTK) a few emulators to run MIDlets on a PC — they present important classes of cell phones (colour/black-white display, different screen sizes,

[19] `http://www.eclipse.org/downloads/`, accessed at 10.10.2007
[20] `http://www.eclipseme.org/`, accessed at 10.10.2007
[21] `http://www.netbeans.org/products/ide/` (accessed at 10.10.2007), please find further details in [Hla06]
[22] `http://www.netbeans.org/products/mobility/`, accessed at 10.10.2007
[23] `http://antenna.sourceforge.net/`, accessed at 10.10.2007
[24] `http://ant.apache.org/`, accessed at 10.10.2007
[25] Find further information on page 24.
[26] `http://maven.apache.org/`, accessed at 10.10.2007
[27] `http://pyx4me.com/pyx4me-maven-plugins/j2me-maven-plugin/introduction.html`, accessed at 10.10.2007
[28] `http://maven.apache.org/plugins/maven-assembly-plugin/`, accessed at 10.10.2007

FIFTH THIRD EX1062
PART 2 of 3
Page 203 of 379

etc.). Nearly every vendor (Nokia, Motorola, Sony-Ericsson, ...) does the same, and provides emulators that simulates their handsets on PC, not only in visual, but often also in behavioural ways. These emulators are essential for the developing process, because they give a first impression of the application without the tedious process of creating a MIDlet-Suite, transferring and installing it on a real mobile phone. Beside these quick-checks, emulators are very useful for debugging or simulations, where more than one cell phone is involved. They are usually very intelligent and allow even to assign different telephone numbers to different instances and send SMS messages from one to another. Some of them also simulate the processing power of the emulated mobile, where calculations take pretty much the same time as they would do on a real phone.

It is often useful to use an emulator for presenting an application to customers, colleagues or testers. Because those people often don't want to bug around with the installation of software they use presumably once, online-emulators[29] are ideal for this purpose. They are either integrated into a web page using *Java-Script* or *Java* and do nearly the same as other emulators do, but with hardly any software requirements. For some emulators it is sufficient to enable Java-Script, if this is not already done, others need an installed *Java Runtime Environment* which is likewise installed on most PCs. A few rules of thumb can be kept in mind, when using (online-) emulators:

- They may look different and behave differently than other emulators or real devices.
- Online-emulators based on Java-Script usually do not support MIDP 2.0.
- Usually resources like SMS, Bluetooth or access to storage devices are not supported and can't be simulated.
- It may be, that users need to install a *Java Runtime Environment* or activate Java-Script in their browser.

Nevertheless, developers must keep in mind, that emulators can't replace real devices — especially for testing and customizing purposes.

Installation

There exist 3 general ways to install the application on your mobile:

1. Surf to a Web- or WAP-page and download the application (usually by clicking an URL that points to a .jad file)
2. Trigger (e.g. your operator's call-centre initiates) a WAP-push which has a similar effect as in 1, except that the user does not have to manually enter a specific URL — the device does it for him/her.
3. Download the application to your PC and transfer it to your mobile via a data-cable or Bluetooth (use *OBEX push*[30]).

[29] For example the *mpowerplayer* (`http://mpowerplayer.com/`, accessed at 10.10.2007).
[30] `http://en.wikipedia.org/wiki/OBEX`, accessed at 15.11.2007

FIFTH THIRD EX1062
PART 2 of 3
Page 204 of 379

J2ME Polish

The most confusing part of this framework is its name, that suggests it has to do anything with Poland. Robert Virkus, the inventor and main developer of the project, clarifies on the project's website:

> "J2ME Polish should mean "turtle wax", at least that's what slashdot reader tod suggests[31]. The name can be confusing really. It has nothing to do with Poland or the language spoken there, rather it should imply that you can polish up your applications with J2ME Polish." [JPa]

The advantages of the project are not only limited to APIs:

- *Resource Assembling*: specify individual resources for different target devices. It's also possible to specify needed capabilities such as "supports MIDP2.0 and screen-size is 176+x208+".
- *Building J2ME applications*: specify target devices, individual .jar files will be created for. The build process supports the phases: pre-processing, resource-assembling, compilation, obfuscation, pre-verification, packaging and .jad file creation.
- The *device database* plays an important role in the above two points. It is used to select devices, based on capabilities specified in the resource assembling, or in the pre-processing phase, to generate individual code.
- The framework supports *CSS like GUI specifications*, which can additionally be enriched with *bitmap-fonts*, that make the application look the same on different handsets. Combined with the resource assembling these are the most powerful advantages of the J2ME-Polish project.
- A speed-optimized *game engine* allows the execution of MIDP 2.0 applications even on MIDP 1.0 devices.
- The *logging framework* makes `System.out.println()` calls visible, even on real devices where they usually are suppressed. Like for other logging-frameworks, the log-level ("debug", "warn", "error") can be chosen.
- Stand-Alone tools, like the *Font-Editor* let developers do their jobs more easily.

Using the framework can be done in two ways: either by using high-level GUI elements of the provided J2ME APIs and compiling it with the J2ME-Polish script, which replaces the elements with *polished* ones. Or, by directly using J2ME-Polish components, what creates apparent dependencies to the framework. The advantage of the first approach is, that developers can decide not to use the framework and stay with the used API. This could be very useful for compatibility reasons, although the link to the produced application is not clear at first sight.

J2ME-Polish is an open source project, but is available in two license flavours:

- Everybody can freely use features of the whole project (find an overview above), but has to publish the resulting code under the *GNU General Public License* (GPL)[32].

[31] `http://developers.slashdot.org/comments.pl?sid=116170&cid=9831734`, accessed at 10.10.2007

[32] `http://www.gnu.org/licenses/gpl-3.0.html`, accessed at 10.10.2007

FIFTH THIRD EX1062
PART 2 of 3
Page 205 of 379

- For commercial publications, where the code needs to be confident, you can buy a commercial license — this option also includes extended support, depending on the bought package.

Although J2ME Polish is an open source project and you are free to extend or modify it where needed, it must be said that it is not an easy task to do so. The code is hard to read, and thus, hard to change or extend, what makes it nearly impossible to adapt it to the developers' needs.

A test for introducing the J2ME-Polish elements to the game template 1 of the *mGBL* project revealed that the required GUI design could not be realized with the given API. Nor was it possible to modify the components to fit our needs, because features and parameters are spread over several classes and it's not clear where, in the code, modifications are necessary. Concretely, a text box of fixed height should have been implemented, which is scrollable when the content does not fit into the box. The API provides a component with similar functionality – a scrollable box that can contain various other elements – but the height can't be limited, unless it fits into the screen. It seems, that the aim of this box is to be as small as possible, but enlarges when needed.

Summary

JavaME might look complicated at the first sight, but has some unique advantages, and therefore makes the race – at least for the application implemented within the *mGBL* project – against other competitors. Reasons, that speak for *JavaME*:

- It is widespread. There is hardly any mobile phone or smart phone that is not able to run a *JavaME* application.
- It is free. Even the most popular development tools are freely available[33].
- *JavaME* has a steep learning curve, especially for developers who are already familiar with Java.
- A lot of API's designed for Java are available in a minimal version that is customized for the limited resources available on mobile devices[34].
- It is customizable. Using the Low-Level-GUI-API, it is possible to make a user interface look whatever you like.

3.2.2 Symbian OS

Symbian OS is a proprietary operating system designed for mobile devices, produced by *Symbian Ltd.* The company exists since 1998, was founded as a cooperation of different mobile phone producers, and the company Psion [Ess06], and is now owned by the companies Nokia (47.9%), Ericsson (15.6%), Sony Ericsson (13.1%), Panasonic (10.5%), Siemens AG (8.4%) and

[33] The IDE's *Eclipse* (`http://www.eclipse.org/`) and *Netbeans* (`http://www.netbeans.org`) are free of charge as well as the needed plug-ins (that is, the *Mobility Pack* for Netbeans, and *EclipseME* for Eclipse).

[34] For example the *kXML* (`http://kxml.org`) XML-reader and -writer API.

FIFTH THIRD EX1062
PART 2 of 3
Page 206 of 379

Samsung (4.5%) [Ess06, Wiki]. Seeing these share rates, it is clear that Nokia is currently the producer with the most Symbian based mobile phones on the market [Ess06].

Symbian defined different version of its operating system, and called them *Series*, where each Series targets different purposes. The *Series 60* (codename "Pearl"), the most spread one, is trimmed to meet the requirements of smart phones, which have a normal cell phone keyboard, and can be operated one-handed. Phones based on the *Series 80* (codename "Crystal"), usually have good organizer functionality, a QWERTY[35] keyboard, a large display, and are equipped with a fully functional e-mail client and web-browser. *Series 90* enabled devices stand out with their large displays and touch screen functionality. Another important series is the *UIQ* (codename "Quartz"), which is designed for devices that are a combination of cell phone and PDA. Again, a large, touchable display and a stylus are characteristic for those devices, which are especially used for professional purposes [Ess06, Wiki]. A prominent exemplar that makes use of the UIQ is Sony-Ericcson's *P1i*[36], which has a built-in two-megapixels-camera, an integrated business card scanner, a web-browser and an e-mail client, to name just a few features.

Developing on Symbian OS

The operating system is not open source, although parts of it are publicly available, to allow developers produce code for the platform. This enables the implementation of development tools for different programming languages, like Python, Visual Basic, Perl and *JavaME* [Wiki]. However, the favourite programming language, still, is C++, because it allows the development of machine-oriented, thus faster, applications, with hardly any restrictions.

Symbian OS C++

The term "Symbian OS C++" refers to the C++ dialect, that is domain specific to the frameworks used to build the operating system itself, and the applications that run on it [HS07].

Symbian want its C++ dialect to be known as different to other C++ dialects for two reasons. First, it is a fact that C++ tends to differ from platform to platform. Second, as the combination of C++ and Symbian has an over 20 years history, it can be considered perfectly adopted to the system's needs.

3.2.3 .Net Compact

> "The .NET Compact Framework is a programming interface and runtime library created as a combination of two Microsoft technologies: Windows CE, an operating system for mobile and embedded 'smart devices', and .NET, Microsoft's reinvention of its programming interfaces and its developer tools." [YD04]

This chapter summarizes information about the host operating system on Windows enabled devices, *Windows CE*, presents basic principles of the *.Net Framework*, and then moves on to its little brother the *.Net Compact Framework*. The latter, of course, is the real topic of this chapter, however the other sections may help to understand how it works.

[35] QWERTY specifies a layout of keys, one is familiar with from typewriters or PC-keyboards (`http://en.wikipedia.org/wiki/Qwerty`).

[36] `http://www.sonyericsson.com/cws/products/mobilephones/overview/p1i?cc=en&lc=UK`

FIFTH THIRD EX1062
PART 2 of 3
Page 207 of 379

Windows CE

The first version of Windows CE was released in 1996. Although it was built with the Win32 API as its core, it shares hardly any code with its heavier desktop pedant. Windows CE has been rewritten from scratch to fit the needs of hardware and software engineers, developing mobile and embedded devices [YD04].

On Windows CE, in contrast to other Windows operating systems, the Win32 interface is still an option for developing applications, and even is in some cases the only possibility to accomplish some certain tasks.

The .Net Framework

The .Net framework is some kind of an umbrella that incorporates interfaces and tools for developers' needs in different divisions and combines them to a larger strategy. Best known are these three application classes: Windows Forms, Web Forms and XML web services.

Some of the main design goals of the framework are [Wikh]:

- Interoperability — because the interaction of new and old applications is often required, the .Net framework provides access to programs that are executed outside the .Net environment.
- Language independence — Microsoft introduced a common type system, where all possible data types are defined. Therefore .Net allows development in different programming languages, and that, again, supports interoperability.
- Security — the .Net framework allows the execution of code at different trust levels, without the need of a sandbox.

To allow the above mentioned language independence, a special architecture has been developed, where one specific programming language is compiled into an independent representation. The *Common Language Infrastructure* (CLI) takes care of this language independent code, and manages further compilation, which makes the code runnable on the target system (figure 3.6).

The .Net Compact Framework

The main goal of the .Net Compact Framework was to enable developers to use a known environment for developing applications for mobile devices. As you continue reading, you may realise, that the correlation between the .Net Framework, and the .Net Compact Framework is similar to the one of Java and *JavaME*.

The .Net Compact Framework was conceived as a subset of the .Net Framework in the full knowledge, that it can't be as extensive as its big brother. But it was designed to reach a maximum consistency between the two frameworks, concerning name spaces, classes, and types. While this makes the framework attractive to developers who are familiar with the .Net Framework, a lot of work has been invested into performance tuning, which should attract former C and C++ developers, too. The Compact Framework team, therefore specifies these design goals [YD04]:

- Build on the benefits of the .Net Framework

FIFTH THIRD EX1062
PART 2 of 3
Page 208 of 379

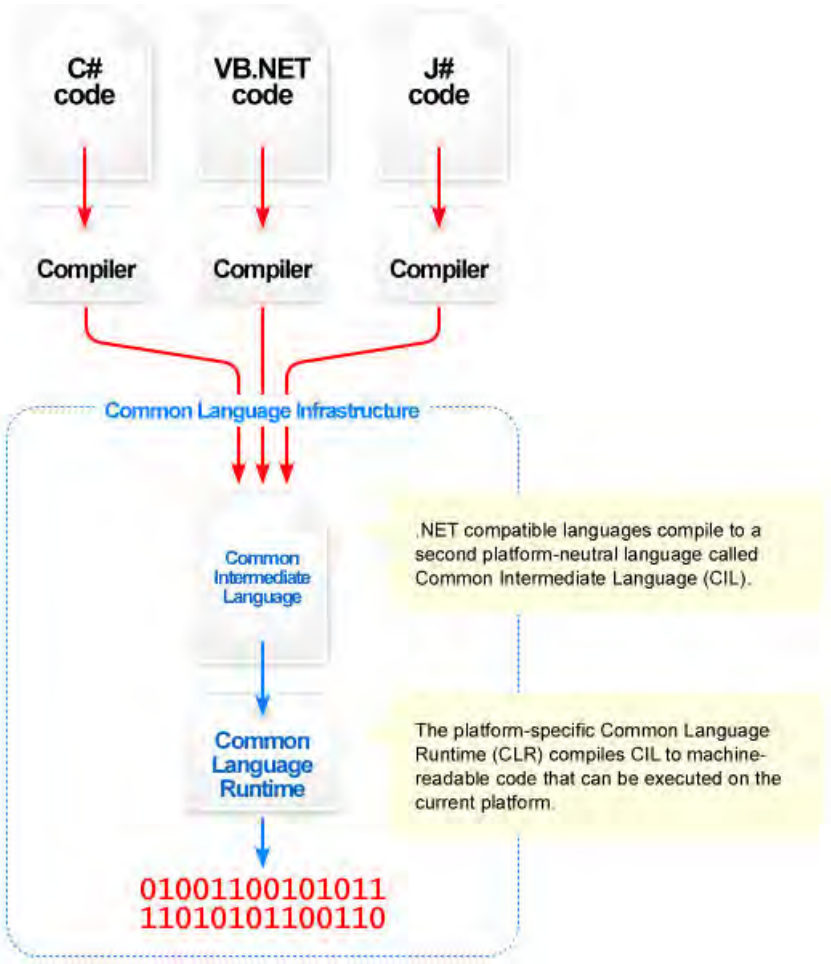


Figure 3.6: Overview of .Net's Common Language Infrastructure (CLI). (source: `http://en.wikipedia.org/wiki/Image:Overview_of_the_Common_Language_Infrastructure.png`)

FIFTH THIRD EX1062
PART 2 of 3
Page 209 of 379

- Maintain consistency with the desktop — In order to scale down the footprint of the framework to 2MB, the team eliminated many of the desktop framework classes, trimmed the remaining class members, and reduced properties, methods and events.
- Run well on mobile and embedded devices — The reduced size of 2MB fits very well to typical Windows CE powered mobile devices with 32-64MB RAM.
- Preserve the look&feel of platforms — To a user, a Compact Framework application is visually indistinguishable from non Framework applications.

Programmers, who are familiar with the .Net Framework, are high qualified candidates to get productive with the Compact Framework. Nevertheless, the significantly reduced feature set makes it essential to allow time to get comfortable with it, since many favourite classes and class members seem to be missing.

Xbox 360

A version of the .Net compact framework is also available for Microsoft's game console *Xbox 360*[37]. This framework is similar to the .Net compact framework, which is optimized for mobile devices, also a subset of the .Net framework, but is optimized for the Xbox 360. [Wikh]

3.2.4 Flash Lite

Adobe's *Flash Lite* is a lightweight version of the Adobe Flash Player, optimized for mobile phones, and other devices with limited resources [Wika].

Flash for mobile devices comes in those three flavours [LJD07]:

- *Flash for Pocket PC* — a version for a PDA, which supports the whole functionality of the Flash 6 and Flash 7 player (including XML, UI components and data loading).
- *Flash Lite* — a profile, especially developed for mobile phones.
- *FlashCast* — a streaming web technology that utilizes the Flash Lite player.

Flash Lite 1.0 and 1.1 are based on the Flash 4 player, include Action Script, but do not support all of the functionality of the desktop profile. It, of course, supports vectors, bitmap graphics MP3 sound and a scripting language and others. However, Flash 2.0 is based on Flash 7, which supports Action Script 2, and should therefore be more familiar to developers who have experiences with Flash desktop applications.

Good news for all Flash developers is, that the *Flash MX 2004 IDE* provides you with everything you need to design, produce and test your Flash Lite application. You can even download different templates for different target devices, that reflect the particular restrictions like the display size, and provide an authentic shape of the mobile phone for the emulation of applications (figure 3.7).

[37] `http://www.xbox.com`

FIFTH THIRD EX1062
PART 2 of 3
Page 210 of 379

Figure 3-2 shows the Flash IDE just after opening a file created using the Nokia 7610 template. You can see the two guide layers that display the phone graphic and specifications. Testing the movie does not export either of these layers, so it is perfectly safe to keep one or both of these in the FLA throughout production.

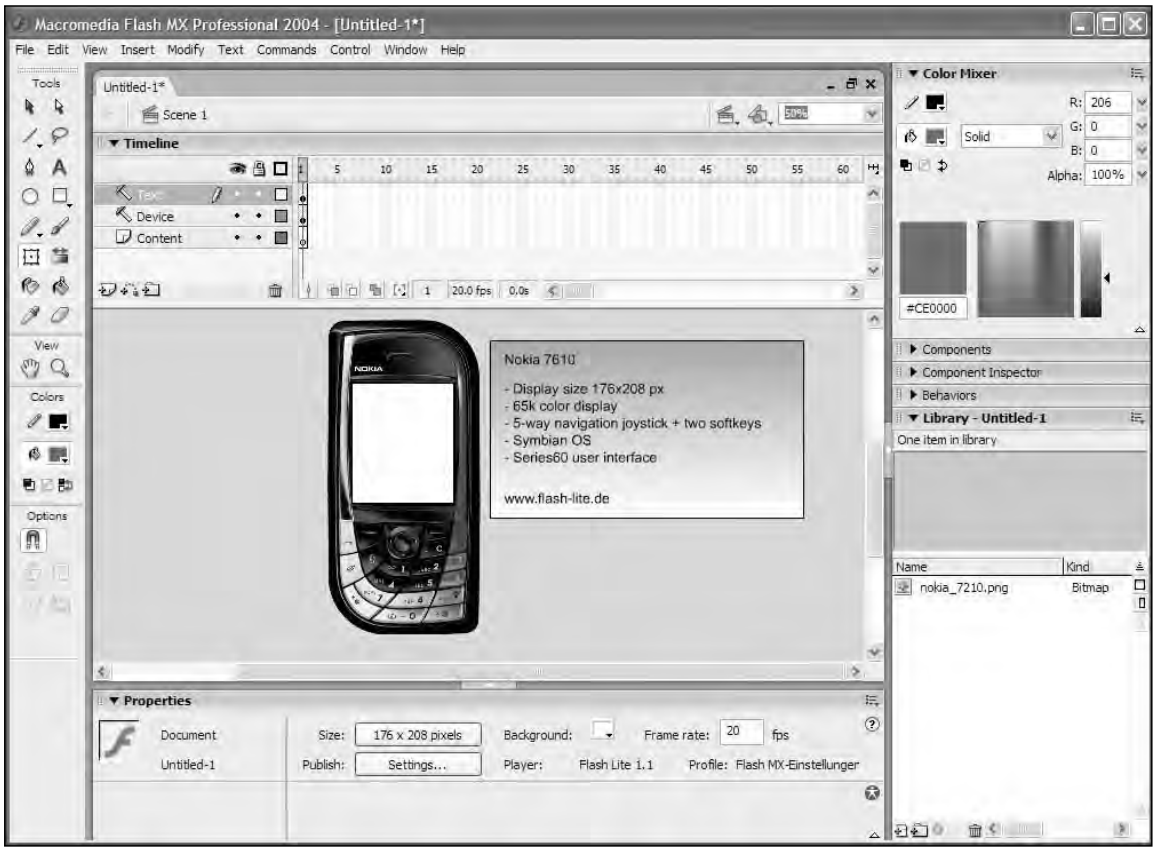


Figure 3-2. The Flash MX 2004 IDE with the Nokia 7610 template

Figure 3.7: Flash MX IDE with the Nokia 7610 Template [LJD07].

Obtaining the Flash Lite player for your device

Now that your Flash Lite studio setup is complete, you might want to think about getting a Flash Lite player for your handsets and any other devices you may own. The Flash Lite player is available for purchase in the software section of the Adobe Store (`www.macromedia.com/store`).

The price is nominal (around $8 per device), and you can begin testing on real devices right away. Each Flash Lite player you purchase from the store requires you to send Adobe your device's 15-digit IMEI number. This number uniquely identifies your device and allows the Flash player to be locked for operation on it. On most handsets, you can obtain this number by typing the following into the keypad:

```
*#06#
```

Disadvantages

Flash Lite applications are not capable of communication via Bluetooth, infra-red, or with a mobile phone's built-in camera [Wika].

The license of the developer suite as well as the Flash Lite player are not very expensive, but also not for free [LJD07].

Advantages

It is quite easy to produce a fancy, graphically affine application, even for people that have few or no programming skills. Therefore, the time-to-market is reduced to a minimum in contrast to competitive products, because the most challenging tasks are quite often design, implementation and especially the adoption of the graphical user interface to different devices.

FIFTH THIRD EX1062
PART 2 of 3
Page 212 of 379

Chapter 4

Mobile Game-Based Learning

As already said at the beginning of this work, *Mobile Game-Based Learning* is an EU project of the 6^{th} framework programme, and was initiated to investigate possibilities to support learning on mobile devices. Therefore, three prototypically game templates, a game management platform and appropriate authoring tools should be specified and implemented. Each of these components will be explained shortly, in the following sections, to give an impression of the whole architecture, concluding with the focus of this work.

4.1 Web-Platform

Figure 4.1 shows a schematic representation of the web platform. Some of the components are integral parts of the platform, whereas others are developed separately. The user management, as well as the OR-mapper (not shown in figure 4.1), which is used to store object-oriented data into a relational database, are such integral parts.

The schematic diagram, figure 4.1, is not complete, but shows all important parts necessary to understand the web platform and the overall architecture.

User Management is spanned through the whole application, because only registered authors are allowed to create or change content. The system handles different types of users, which inherit different access rights, because learners also have to have access to some parts of the platform to download and manage their games or change personal settings, like user-name and password.

Game Management inherits the creation of games by authorized users, as well as the management of who and when people are allowed to download and use a game. It is possible to assign single users, or whole groups to a particular game. Additionally, a start and end date can be specified, within the before mentioned users are allowed to download the game, and let it communicate with the server.

The Game Selection Tool helps authors to decide for a particular type of game, by specifying their requirements and constraints. Currently, this tool is stand-alone, in the way that it has no direct connection to any game authoring tool. Instead it serves as advisor, that can be used to pick a game outside of the *mGBL* context. When, some time, more game templates are available, the tool can be seamlessly integrated into the game creation process.

FIFTH THIRD EX1062
PART 2 of 3
Page 213 of 379

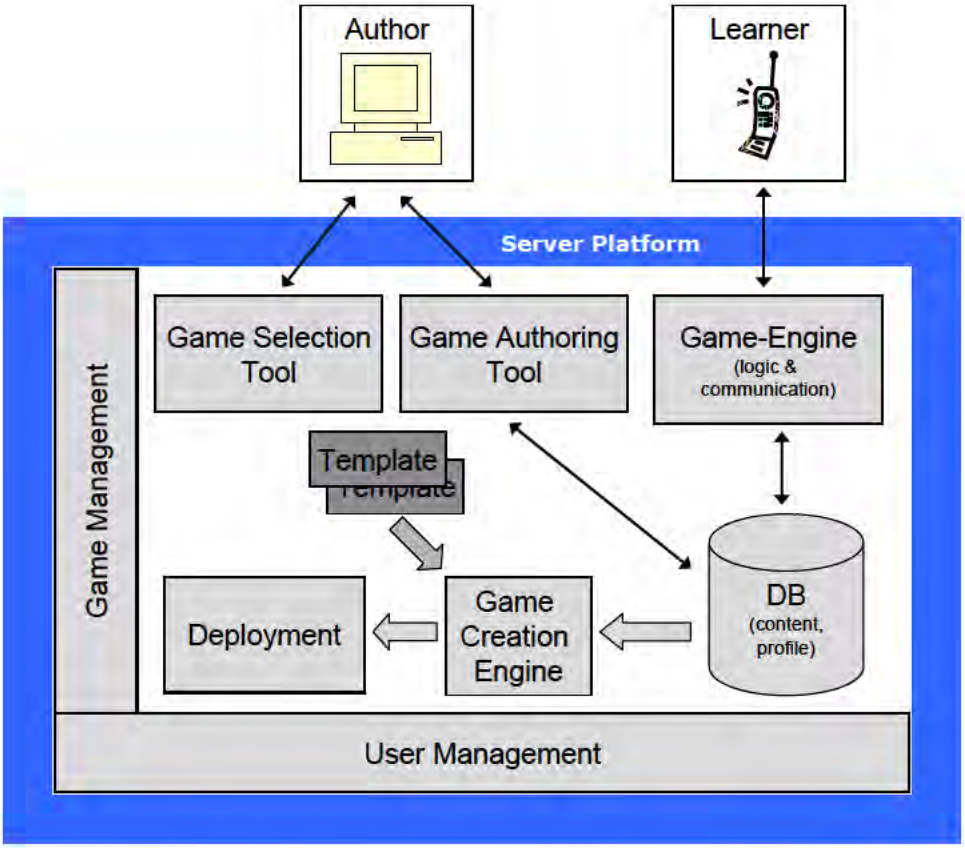


Figure 4.1: Overview of the web platform, showing the most important components

The Game Authoring Tool is separately developed for each game template, and tailored to each specific need. Therefore, it should rather be called "authoring tools"[1]. One such tool is directly and transparently connected to the game template it is implemented for. Therefore the introduction of a new game template requires its connection with the appropriate authoring tool.

4.2 Templates

The *mGBL* project demands at least three different game template prototypes as one of its outcomes.

During the authoring process, the author selects one of the implemented templates and creates content explicitly for that template. Both are combined by the system, and are compiled to one playable game (or game component, see subsection 5.1.1). Actually, the web platform produces not just a game, but several instances of the same game, that are (potentially) tailored to different devices (see section 4.3 for details).

4.3 Distribution

The deployment component, sketched in figure 4.1, is responsible for creating different versions of one game, which are tailored to different target devices. This may be necessary for some game templates, because they are designed for one, or a few, particular handsets, and need to be "transformed" to fit other devices[2].

As described in section 4.1, the so created different versions of one game can be downloaded via the web platform, by authorized users.

4.4 Communication

For a better readability, a so called *Asynchronous Communication Component* has been left out in figure 4.1. Therefore, for SMS-based, thus asynchronous connections, a handset does not directly communicate with the Game Engine, but with the Asynchronous Communication Component. It queues incoming requests, and schedules their processing by the Game Engine. The other way around, it queues outgoing messages and reliably sends them to the target handsets.

4.5 Roles & Tasks

Talking about a game template or the web-platform, the term user arises quite often. Keep in mind, that this term can be used when talking about an author, a player, or a validator.

[1] Find the design of the authoring interface for game template 1 in appendix C.

[2] It, indeed, would enrich the possibilities of *Ahead of the Game*, because such a tool would enable us to adopt the game to many different mobile phones, by just creating a few, slightly different setting files (including for example graphics in different resolutions). However, this step has not been taken so far.

FIFTH THIRD EX1062
PART 2 of 3
Page 215 of 379

Simply speaking, a user is someone who uses the software. Following, we define three roles a user can take.

4.5.1 Roles

A game author can be a teacher or a student. In principle everybody can author a game, but if games are used in learning groups, they need to be validated by a game validator, who is usually the teacher. Games are usually played by students (game player), nevertheless teachers may want to verify their game by playing it themselves. However, it is unclear, yet, if an author can play his/her games via an emulator or participates the competition like other learners (see table 4.2). Table 4.1 gives an overview of the most important roles[3].

Table 4.1: Different user-roles.

	Author	Validator	Player
Teacher	x	x	
Student	x		x

4.5.2 Tasks

As shown in table 4.2, the tasks of a teacher are authoring and validating a new game, as well as administering learner-groups. The major task of a student is playing a learning game. The student/player can also view and alter his/her personal profile and inspect his/her learning progress.

Table 4.2: Tasks of the most important roles.

	Teacher	Student
Authoring	x	x
Validating	x	
Playing a game	o[4]	x
Creating and maintaining one's own profile	x	x
Viewing one's learning progress	x	x
Administering a learning group	x	

4.5.3 Server-side & Client-side Tasks

All games are authored on the server. For each game type a specific authoring interface is available. The authored game must be validated by an authorized validator, who is typically

[3] Other roles may be useful for the web-platform (e.g. the role "Administrator"), but are neglected here.

[4] Teachers may want to verify their games by playing them themselves, but it is unclear, now, how this will be supported.

FIFTH THIRD EX1062
PART 2 of 3
Page 216 of 379

the teacher. Administering of learning groups also takes place at the server. Moreover, the students/learners create, view and maintain their profiles, as well as view their overall learning progress on the server. Table 4.3 summarizes all important activities, while focusing on the two platforms, unlike table 4.2, which shows the same tasks in contrast of different users.

The client application is mainly a framework for playing the authored game. However, the user's full profile, containing (beside other data) scores of all played games, is stored and maintained on the server.

Table 4.3: Tasks in contrast of the two platforms.

	Server	Client
Authoring	x	
Validating	x	
Playing a game		x
Creating and maintaining one's own profile	x	
Viewing one's learning progress	x	x
Administering a learning group	x	

4.6 Focus

This work focuses on the specification and implementation of one game template — *Ahead of the Game*. Although details of the web-platform will be mentioned in the next chapters, it is just for the reader's better understanding of the whole picture.

In the next chapters, the specification of the two components of *Ahead of the Game* is shown, including their game flow and interaction with each other. The specification concludes with the game's scoring concept, controls, and non-features, that are features that might be associated with the application, but are not going to be implemented for certain reasons[5].

The following chapter (chapter 6), explains how the specification is implemented. It starts with the used technology and a package overview, before explaining the underlying software design. Based on the data model, the basic functionality and potential of the application is explained, which is rounded off by an overview of important settings to configure the application. The implementation chapter concludes with other interesting aspects, like the server communication, file structures, user profiles, and possible extension points for future implementations.

[5]See section 5.5 for more details.

FIFTH THIRD EX1062
PART 2 of 3
Page 217 of 379

Chapter 5

Template Specification

5.1 Overview

This chapter gives you an overview of the game template 1 *Ahead of the Game*. First I will explain in some simple statements how the game-flow works, and continue with more details.

5.1.1 Components

There are two major components — the quiz (*Fastest First!*) and the simulation component (*Crisis!*). While the first is used for testing the user's knowledge, the latter is used to put him/her into a virtual emergency scenario (e.g. a car crash for e-health, or an imminent inflation for e-commerce). The quiz component lets the user know if s/he has done right and how much points s/he has gained, for each question, where in contrary the simulation – like in real life – does not give immediate feedback. Although, if the user asks for, s/he gets detailed feedback at game end, via the web-platform. The major idea behind those two components is that the one is used to teach the user knowledge that s/he has to apply at the simulation phase.

5.1.2 Game-Flow

Figure 5.1 gives an overview of how the game is composed of the two components, quiz and simulation. The quiz component is played first, where the player gets questions in sequential order, concluded by an upload of the scores to the web-platform. The simulation has a more cyclic nature, where the user returns to the same, but changed (e.g. deteriorated), situation. Before the score of this component gets transferred to the server, too, the user has to assess his/her own doing.

Additionally, figure 5.2 repeats this presentation in a more visible way, enriched with more details according both components. In this schematic representation, the sequential order of quiz questions and simulation situations is more obvious.

The two game components can be authored/played independently, which results in the following possibilities:

- Only quiz
- Only simulation
- First quiz, then simulation

FIFTH THIRD EX1062
PART 2 of 3
Page 218 of 379

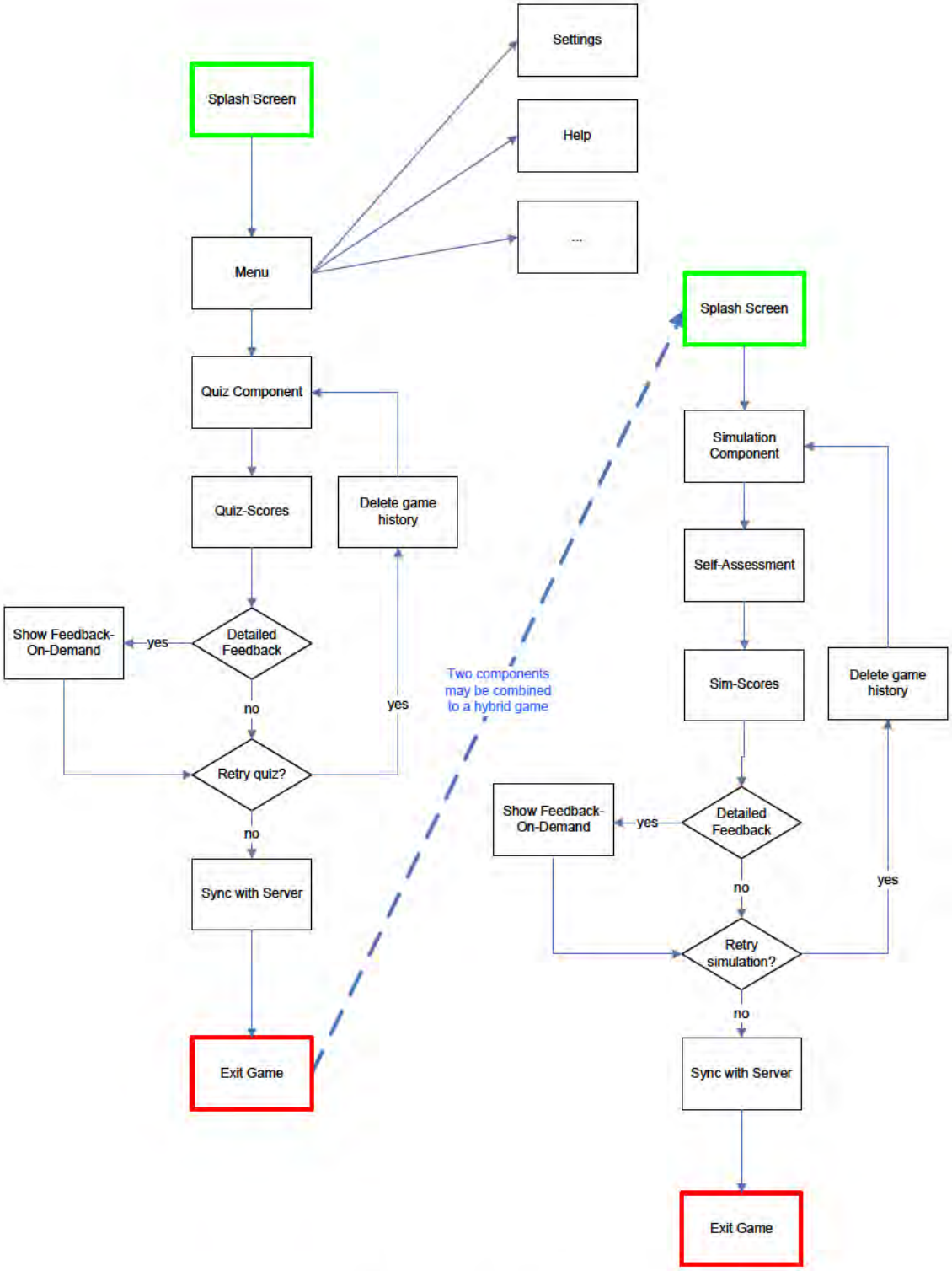


Figure 5.1: Game flow overview

FIFTH THIRD EX1062
PART 2 of 3
Page 219 of 379

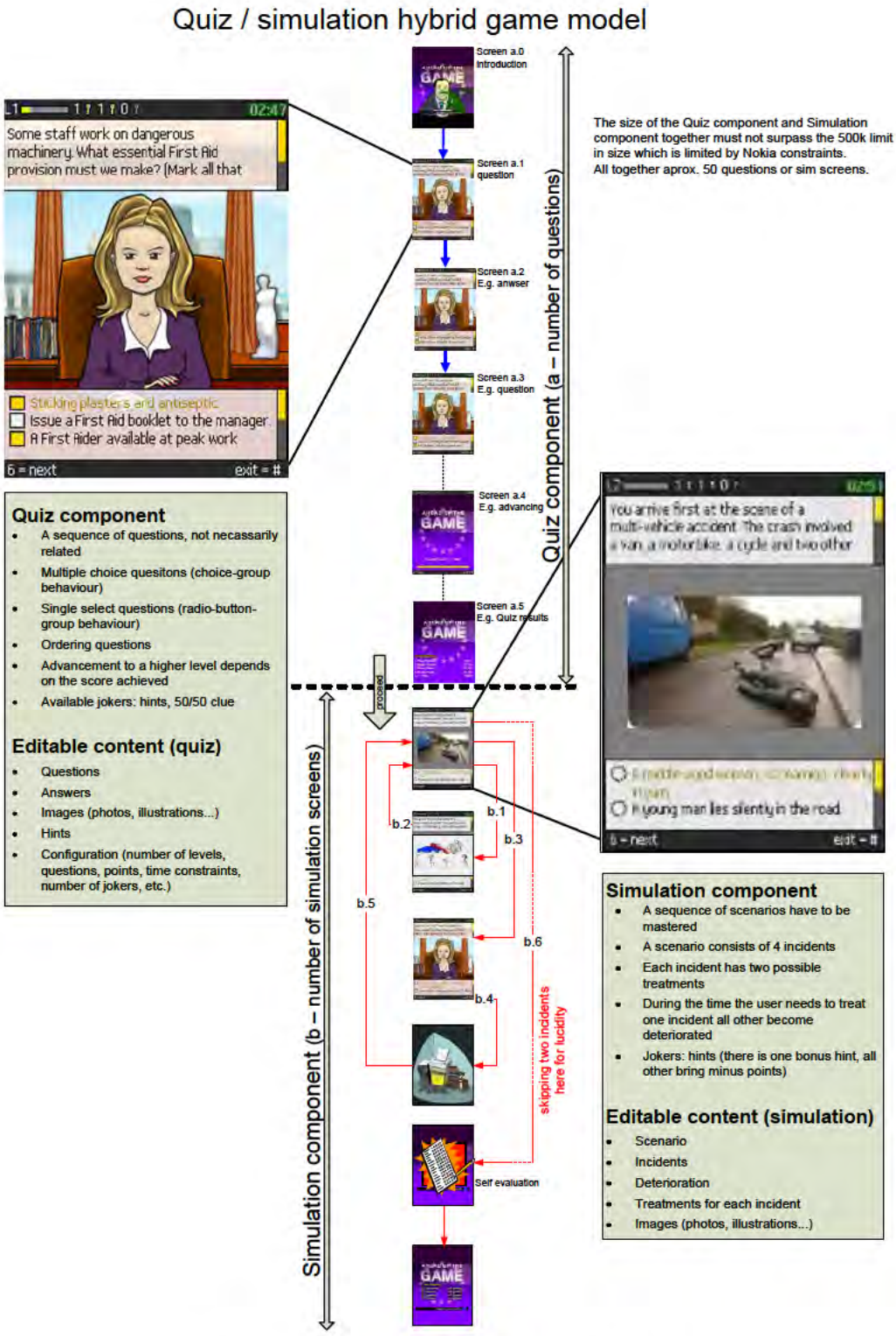


Figure 5.2: Overview of the whole game

FIFTH THIRD EX1062
PART 2 of 3
Page 220 of 379

5.1.3 Game Levels

Both, the quiz and the simulation component are split up – or may be split up – into several game-levels, for example, to separate different topic- or skill-levels. The technical realization, though, differs in that point from the specification to support more flexibility (see chapter 6 for more details).

5.1.4 Points

User-obtainable points can always be positive or negative and their amount can be freely chosen by the author (as long as they are whole-numbers). The exact implementation of calculating the points is specified in section 5.4.

5.1.5 Progression

In both components the user gets confronted with all CeLOs sequentially, as they have been authored. The author can choose whether to randomly rearrange the different answer-options of one question, or not (in this case they appear in the order as they were authored).

5.1.6 Game Saves

Every time a CeLO has been completely processed it will be saved to the mobile phone's internal memory. When the player exits the game, not having completed a question or a simulation, s/he has to do it again.

At game end the user has the chance to improve his/her score by playing another component of the same type. When the player is satisfied with his/her score the system manages the upload of relevant data to the server (achieved points, used jokers, etc.).

5.1.7 The Boss

At the beginning of the game, the user has to select his/her boss gender by visually pick one of two avatars (a female and a male boss). This avatar will be used to present to players the game challenges. It's imaginable that in a later stadium of the project a more extensive collection of avatars will be available, which underlines the playful character of the game template.

To build an indirect multi-player game, it was planned to position a nameplate on the boss' desk, where the name can change. Thus, at the beginning of each level, the current high-scores are retrieved from the server, to use the top-ranked competitor's name as the bosses name. While adding information of the current status of the ranking, it creates also competition, and incites the player to get a good score. However, this feature has not been implemented, yet, because of tight time constraints.

5.2 Component 1 — Quiz

5.2.1 Principle of game play

As shown in figure 5.3, the application can be paused (exit before game ended). That is checked first, at every game start. The game starts, either at the first game-level with the first question,

FIFTH THIRD EX1062
PART 2 of 3
Page 221 of 379

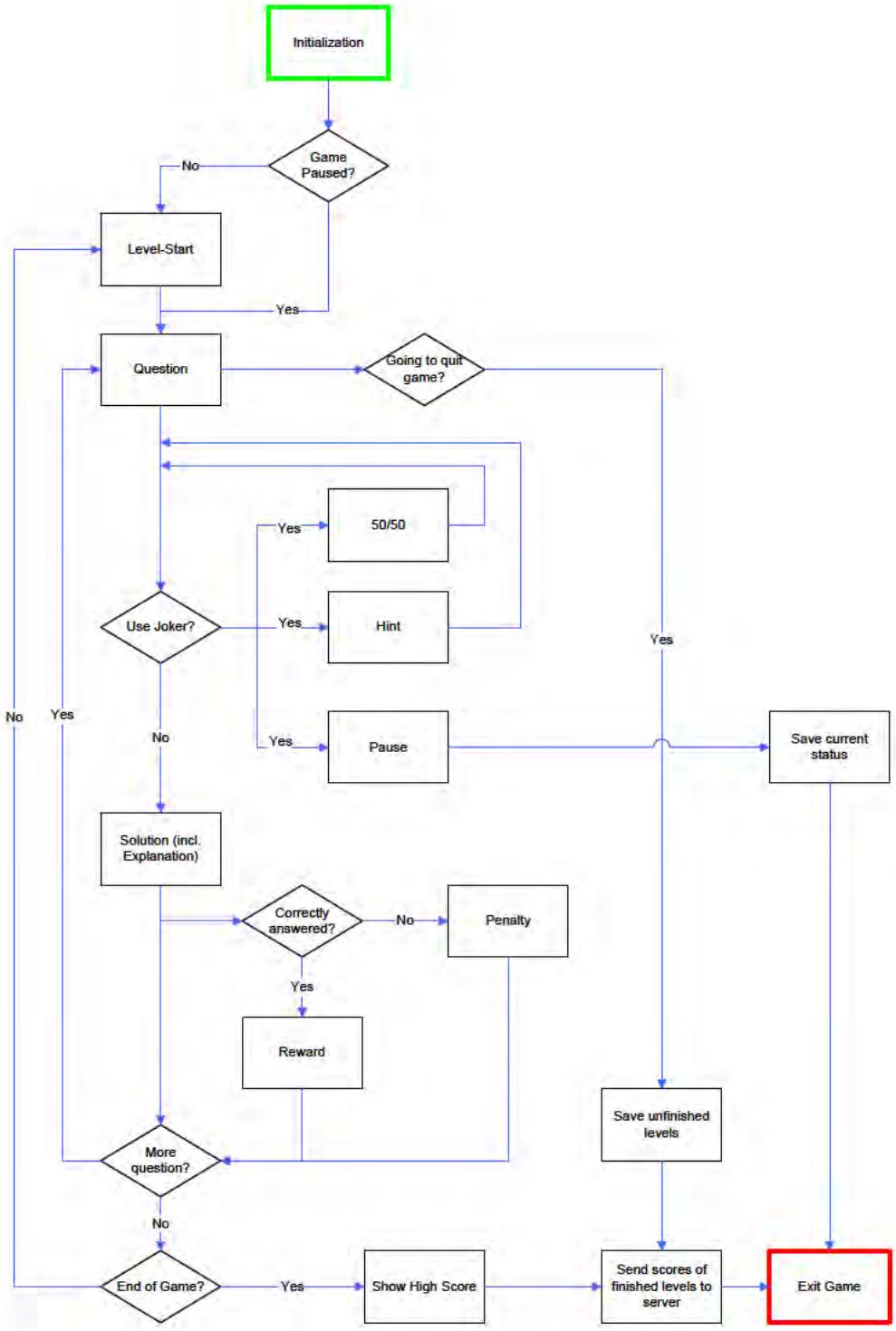


Figure 5.3: Gameplay of the quiz component

FIFTH THIRD EX1062
PART 2 of 3
Page 222 of 379

or at the saved position. While a question is presented to the user, s/he might use a joker, if allowed to. A joker is permitted, if the user has one or more left, and it is appropriate — for example a 50/50 joker is not appropriate for ordering-questions. When the user locked the answer-options, s/he gets feedback about their correctness, as well as an explanation of wrong ones. When the last question of the last game-level has been answered, the user's score is transferred to the server, which responds with a high-score-list that is shown to the user. On every game exit – at game end, or precipitate – the current game status is saved to the mobile phone's internal memory.

5.2.2 Jokers

In the quiz component the user is supported by three different types of jokers:

- 50/50 clue (can only be used with Single-Click questions)
- Hint (brings the user bonus or penalty points)
- Game pause

The author specifies how many of each joker-type the user can request during the game.

5.2.3 Feedback

The feedback is immediate — the user sees which answer-option was correct, which not and gets a short explanation why they are right or wrong. After the game, the player can obtain detailed feedback in respect of all the answer options.

5.2.4 Rewards/Penalties

The nature of the rewards and penalties makes it mandatory that the user can only earn one exemplar of each. Therefore the user earns one exemplar only at special occasions (when s/he gains a lot of points in relation to the maximum possible score).

Reward examples:

- Medal (e.g. "head of coffee machine")
- A sarcophagus plant (for new office)
- Mow the boss' lawn
- Cellular phone to be reachable even at night

Penalty examples:

- shred financial accounting
- go shopping with the boss' wife (a lot of bags)
- act as clown at the next company party

Both, penalties and rewards have been intentionally designed with a bad touch, to keep the player's tension and motivation high.

FIFTH THIRD EX1062
PART 2 of 3
Page 223 of 379

5.3 Component 2 — Simulation

5.3.1 Principle of game play

As illustrated in figure 5.4, the user enters a simulation scenario, where s/he has to decide which of 4 critical cases to treat first (single click question). The prioritised case must then be "treated" by answering a second single click question. The default situation here is that there are only two treatment options per case, and the user will get no immediate feedback (see subsection 5.3.3). However, the author may decide to allocate more than 2 treatment options on a case by case basis.

After the user has decided and treated the first of the 4 cases, s/he returns to the remaining 3, which have meanwhile changed, e.g. deteriorated. It may even now happen that one of them "expires", which is best explained with a health scenario, where patients really could die. When we are not talking about people but about e.g. stock values, the case could be a stock that falls beyond a critical level.

The user has to take care of all 4 cases, prioritising at each stage. When all cases have been treated or have expired, the scenario ends and a new simulation may be requested.

5.3.2 Self-Evaluation

At the end of this component the user is asked to assess his/her progress (poor, fair, good, excellent), and gets a feedback whether this is true or not (e.g. "you are /not /almost/ right — your assessment: excellent, your score: good"). S/he then gets access to actual points scored. These steps are illustrated at the bottom of figure 5.4.

5.3.3 Feedback/Explanation on Demand

There is no instant feedback like in the quiz component (see subsection 5.2.3). The user rather has to cope with the simulation situation as it develops: the developments in the simulation represent the kind of feedback, provided by developments in real-life situations. However, the player can ask for detailed feedback at the end of the component, via the web-platform.

5.3.4 Jokers

On the one hand, no 50/50 clues are allowed (or necessary) because a simulation scenario consists of four cases, where none of the player choices (priorities, choice of treatment options) is absolutely wrong (it's more a question of what ought to be their priorities). On the other hand, some hints do make sense, and that's why one bonus-hint exists that brings bonus points for using it in the right situation, but all other hints bring minus points for using them.

5.3.5 Rewards and Penalties

Rewards and penalties work the same way as in the quiz component, please see subsection 5.2.4 for more details.

FIFTH THIRD EX1062
PART 2 of 3
Page 224 of 379

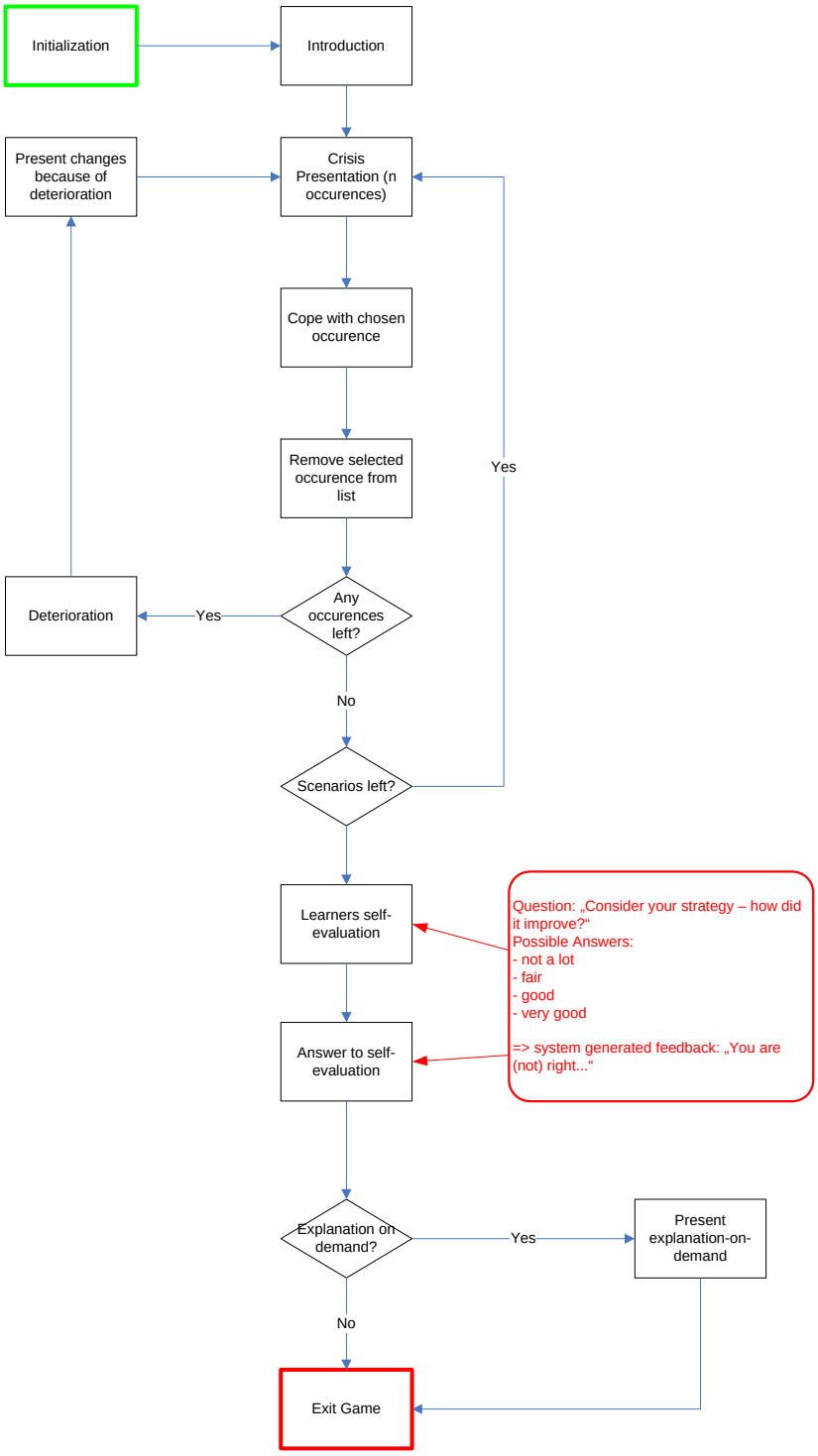


Figure 5.4: Gameplay of the simulation component

5.3.6 Authoring Example

Please find an example of how a simulation component may be authored in table 5.1. For a better readability no cells for points or timing issues have been reserved.

FIFTH THIRD EX1062
PART 2 of 3
Page 226 of 379

Table 5.1: Simulation authoring example

Element / Time	t_1	t_2	t_3	t_4
Simulation description	You arrive first at a multiple car accident. What do you do first?	Who are you going to treat next?	Two people saved, who is next?	Last treatment...
Hint	Prioritise the life-threatening condition.	Prioritise the life-threatening condition.	Prioritise the life-threatening condition.	
Case 1	Man with head-wound	Man with bleeding head-wound	Man with bleeding head-wound	Man with bleeding head-wound
Priority	2	1	1	1
Case 2	Pregnant woman	Pregnant woman	Pregnant woman, screaming	Pregnant woman, screaming
Priority	4	4	3	3
Case 3	Boy sitting silent on the sidewalk	Boy sitting silent on the sidewalk	Boy lying silent on the sidewalk	Boy lying silent on the sidewalk
Priority	2	2	1	1
Case 4	Unconscious man	[The man died]		
Priority	1			

FIFTH THIRD EX1062
PART 2 of 3
Page 227 of 379

Simulation authoring example (continued)

Element / Time	t_1	t_2	t_3	t_4
Question for case 1	How do you treat the head-wound?	How do you treat the bleeding headwound?	How do you treat the bleeding headwound?	How do you treat the bleeding headwound?
Answer 1	Bandage	Compression bandage	Compression bandage	Compression bandage
Answer 2	Cleanse	Patch	Patch	Patch
Question for case 2	What's best for a pregnant woman?	What's best for a pregnant woman?	What's best for a pregnant woman?	What's best for a pregnant woman?
Answer 1	Lay down, feet up	Lay down, feet up	Lay down, feet up	Lay down, feet up
Answer 2	Sports	Sports	Sports	Sports
Question for case 3	A silent child - what do you do?	A silent child - what do you do?	A maybe unconscious child - what to do?	A maybe unconscious child - what to do?
Answer 1	Give him candy	Give him candy	Start a conversation	Start a conversation
Answer 2	Start a conversation	Start a conversation	Check the pulse	Check the pulse
Question for case 4	What's essential with unconscious ones?	[The man died]		
Answer 1	Free the airways			
Answer 2	Be very silent			

FIFTH THIRD EX1062
PART 2 of 3
Page 228 of 379

5.4 Scoring

This section describes the different scoring models according to the already described specifications (see sections 5.2 and 5.3).

Points can always be positive or negative, and their amount can be freely chosen by the author (as long as they are whole-numbers). The exact implementation of calculating the points depends on the specific implementation (quiz or simulation, see subsections 5.4.1 and 5.4.2).

5.4.1 Quiz

First of all, it should be clear that every question may bring plus or minus points, depending on the correctness of each answer-option, the used time and used jokers. The basic formula to calculate the points is:

$$P_{Question} = \sum_{Answer=1}^{4} (P_{Answer} + P_{Hint})$$

There is a switch in the game configuration that activates a time-based point calculation, which expands the formula to:

$$P_{Question} = \sum_{Answer=1}^{4} \left(P_{Answer} * \frac{t_{available} - t_{used}}{t_{available}} + P_{Hint} \right)$$

It's also possible to activate a minimum level of points, by setting a *Minimum Time Factor* (between 0 and 1, as a floating point value). Setting the factor to 0.25, for example, tells the system that one quarter of the points will be gained by the user, if the question has been correctly answered but the time has run out. A factor of 0 tells the system to give no points when time's up, and 1 equals setting the time-based points calculation to false.

Single-Click

The above defined formula is sufficient for Single-Click questions.

Multiple-Click

As for Multiple-Click questions mostly two or more answer options are correct, and to avoid cheating, positive points are assigned only when all answer options are correctly checked (as they were authored to be correct).

Answer-Ordering

Each answer option can be assigned points (as for the other two types), so the user earns points for every correct position of an answer option. That means: if an option is authored to be for example on the 3^{rd} position, the user gets points if s/he places it correctly. For that specific item s/he may get 5 points, whereas for placing the 1^{st} item correctly s/he gains only 1 point. The reason for that is to be found in the authoring, because the author can freely specify the amount of points[1] for right or wrong answer-options.

[1] Points have to be specified as whole numbers, but can be positive or negative.

FIFTH THIRD EX1062
PART 2 of 3
Page 229 of 379

5.4.2 Simulation

In a similar way to some of the questions in the quiz component the user has to answer single-click questions:

1. Prioritise among cases: select ONE (earns points)
2. Choose a treatment option relating to the prioritised case: select ONE. Although here with only two possible answers there is a fifty-fifty chance for the user, the response also brings points according to the used time.

The basic formula here is:

$$P_{Scenario} = \sum_{Incident=1}^{4} (P_{Question} + P_{Incident} + P_{Hint})$$

Where $P_{Question}$ is taken from the quiz component's formula in subsection 5.4.1.

5.4.3 Rewards & Penalties

Users do not earn points for rewards or penalties. For more details on rewards and penalties see subsection 5.2.4 on page 42.

In advanced versions of the game, rewards could be extended by the possibility to reveal hidden material (CeLOs) to gain extra points. This idea is described in more detail in section 7.3 on page 71.

5.4.4 Hints

Hints may bring the user plus, minus or zero points. Again, the amount of points is up to the author.

5.4.5 Time-Aspect

Depending on the game's configuration, time is not only noticeable for giving answers, but also for gaining points. Setting the parameter `ANSWER_POINTS_TIME_BASED` to true, lets fast users gain more points than thoughtful ones, by taking the used time into account for the calculation of points.

5.4.6 Authoring

As mentioned before, the amount of points that can be gained by the user is not limited by the game template. However, the authoring interface could be implemented in the way that it is, for example, not possible to assign more than 2 points for a hint, while points for answer-options are limited to -3 to 8.

5.5 Non-Features

The aim of this section is to make clear what features will not be implemented, and why. Table 5.2 lists the most important of those features, someone might expect to be part of the application, and the reasons for not implementing them.

FIFTH THIRD EX1062
PART 2 of 3
Page 230 of 379

Table 5.2: Non-Features, that is, features someone may expect to be implemented, but won't for the described reasons.

Feature	Comment
Previous screen	There is no use case for letting the user access the previous screen/CeLO, and therefore it will not be implemented.
Support all available handsets	Although the defined minimum requirements are met by a wide range of mobile phones, we picked only a few of them for a prototypically implementation (see subsection 6.1.2), however, there still are a lot of differences between those devices.
Video support	Playback of video-snippets itself is not easy. Additionally it raises up a lot of other questions like "what (should) happen(s) if a device does not have (certain) video capabilities" or "videos reserve a fair amount of memory within a jar-file, that not all phones can cope with. Whereas streaming also is not an option because it still does not work well for mobile devices".

5.6 Controls

Figure 5.5 shows the keyboard layout for mobile phones without touch-screen. However, smartphones with a touch-screen, which are quite popular now, are also supported by the application. The player can use both: his/her stylus ,like s/he is used to, and the keyboard like it's defined in figure 5.5.

FIFTH THIRD EX1062
PART 2 of 3
Page 231 of 379

4 Controls

The controls layout on the user's device is (independent of the component):

Controls	
	2 , ▲ Scroll up 8 , ▼ Scroll down 5 , ● Select 6 , ▶ Next 4 , ◀ Joker selection ★ Menu

Figure 5.5: The controls layout on the user's device (independent of the game component)

Chapter 6

Template Implementation

6.1 Standards

6.1.1 API

As there is hardly any actual mobile phone that does not support it, and there come a lot of benefits with it, the chosen API is the *Java 2 Micro Edition*, based on MIDP2.0 and CLDC1.1.

6.1.2 Devices[1]

For commercial game developers there exist lists that mention about 400 different mobile devices that should be supported by an application. Because we lack the financial power of commercial game producers, we concentrate on just a few mobile phones:

- Sony-Ericsson M600i
- Sony-Ericsson P990i
- Sony-Ericsson v630i
- Nokia 6230i

Those devices are representatives for classes of mobile phones with different screen resolutions. Therefore, developing the application for those devices should enable us to support also other devices of those classes.

6.1.3 Screen Sizes

Game Template 1 *Ahead of the Game* will be optimized for the following selected screen sizes, that result from the above specified test-devices:

- 176x220
- 208x208
- 240x320

[1] A detailed description of the devices' features can be found here: http://www.gsmarena.com or here: http://www.j2mepolish.org/devices-overview.html

FIFTH THIRD EX1062
PART 2 of 3
Page 233 of 379

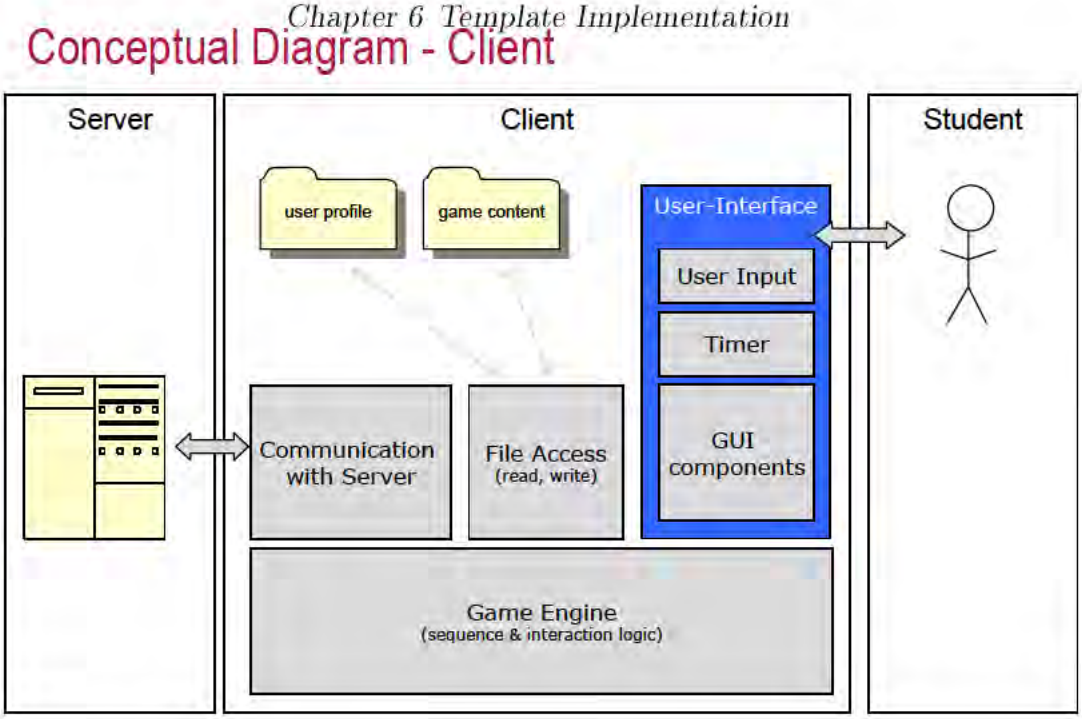


Figure 6.1: Overview of the whole system from a client's perspective

6.1.4 Multimedia Content

The playback of multimedia content like video and sound clips requires the Multimedia-API (MMAPI), which still is no guarantee that a certain format is supported. Therefore supporting multimedia content also asks for alternatives in cases where the playback of the content is not possible. Thinking of a preparation video that explains a current situation, a picture or a text is often not an option.

In view of all these problems we are not going to support multimedia content yet, but will try to keep the software design open for later implementations.

6.2 Architecture Overview

As portrayed in figure 6.3, that, in contrast to figure 4.1, focuses on template aspects, the author creates content using the web-platform (*Web-Site*). The so created data is stored in a central database (*DB*). The system automatically accesses the database, incorporates the content with the game template (*Game Creation Engine*) and makes the game available for download by students (*Deployment*). A separate component handles subsequent communication with the client handsets (*Game-Engine*).

Figure 6.2 shows the concept of how the game template is filled with content. The author creates a game concept that mainly consists of texts and graphics. This data is authored using, and stored in a database provided by, the web authoring tool. A pre-processor and compiling tool then incorporates the content provided by the author and the game template to an executable file.

In addition to figure 6.3 and figure 6.2, a combined view of them is presented in figure 6.1,

FIFTH THIRD EX1062
PART 2 of 3
Page 234 of 379

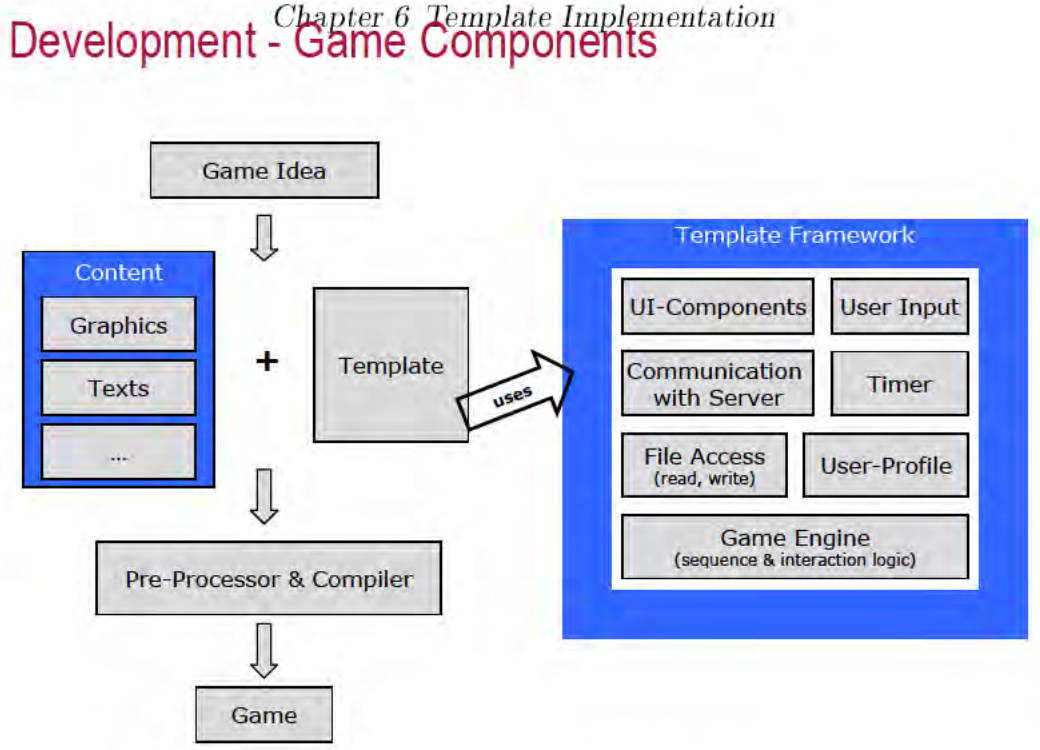


Figure 6.2: Schematic representation of the game creation process

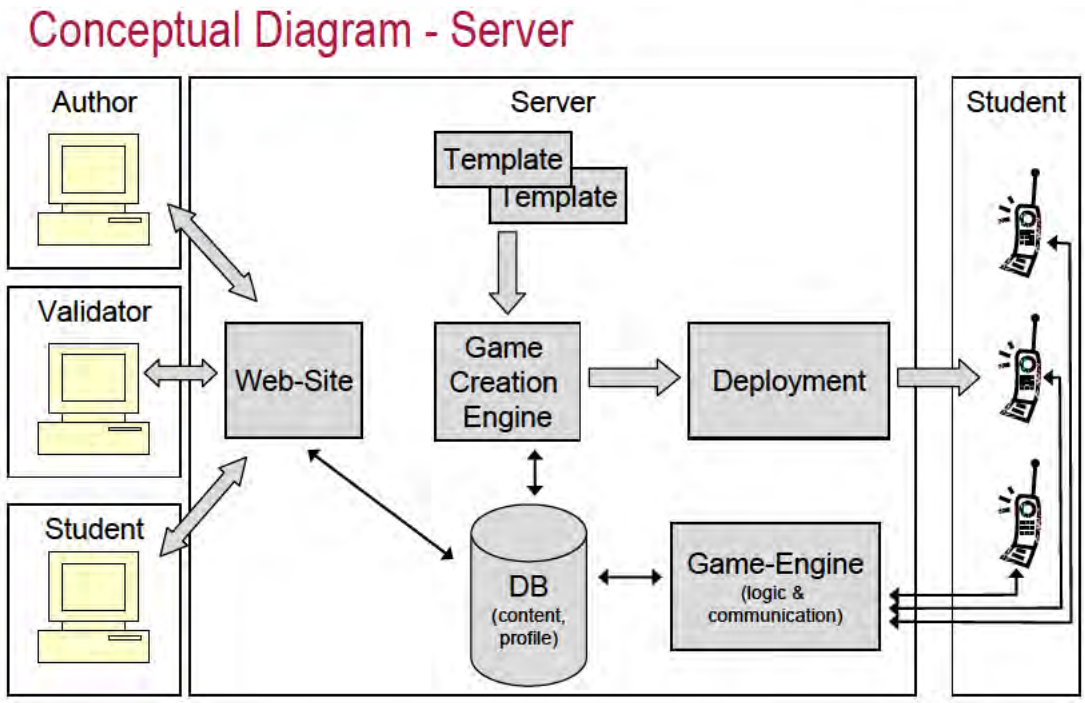


Figure 6.3: Overview of the whole system from a server's perspective

to highlight essential components of the game template in relation to the outside world.

6.3 Package Structure

The main package is `at.researchstudio.sat.mgbl`; subpackages are described in table 6.1.

Table 6.1: The application's package structure

Package	Description
`io`	Classes of the IO package take care of data-communication with the server. These connections are based on either SMS or HTTP.
`model`	The model classes mainly represent the CeLOs. In some cases, when it is necessary to store additional data like achieved points, instances of these classes get updated from within view or controller classes (e.g. `QuestionRepresentation`).
`model.userprofile`	The user's profile is, at game end, transferred to the server, where it's used to compute statistics including other games and game instances. The profile is also used in-game, to feed the action trigger, the self-assessment component, and others.
`store.dao.impl`	There are two main purposes of this package: reading and writing data to/from the database(s). A huge part is used for reading the game content from the provided data.xml file. Another, in contrast very small, part is used to write essential data to the mobile phone's RMS-Storage (to be used later for statistical purposes).
`store.dao.impl.gamedata`	All classes of this package are used to retrieve CeLO information from the used database (the data.xml file). The entry point for XML parsing is the class `AheadOfTheGameParser`, which initiates the XML parser, reads some general data, skips all data of previously solved, or future game levels, and finally passes the parser instance to other objects, which do the CeLO parsing.
`ui`	The package is dominated by subclasses of `AbstractRepresentation`, which support basic functionality for drawing content on the mobile phone's screen, and reacting on key or pointer events.

FIFTH THIRD EX1062
PART 2 of 3
Page 236 of 379

The application's package structure (continued)

Package	Description
`ui.framework`	The UI-Framework consists of elements needed to create a screen that is presented to the user. The most important elements are: • Scrollbar • Textbox • Checkbox-Group • Radiobutton-Group
`util`	This package provides, beside other classes, utils for mathematical, graphical and timing purposes. Additionally to those helpful functionality, the class `Properties` stores publicly accessible constants that are used to configure the application. These are for example constants describing the text-boxes' height and width, text snippets, boolean values to activate the debug mode or to shuffle answer options, and so forth.

6.4 Software Design

Figure 6.4 shows the most important packages and their elements of the client application. These elements are described in more details in the following sections.

6.4.1 Controller

The Controller is the central element of the application (on the left side in figure 6.4). It triggers the loading of new content, manages all loaded CeLO elements, and takes care of presenting each CeLO to the user.

Key-/pointer events

Keystrokes and pointer events are also handled by the Controller, if they are system-relevant (e.g. opening the menu); or they are forwarded to the current representation, which itself may forward it to one of its components.

Draw, Repaint

Whenever the MIDlet is called to draw the screen content, it forwards this call to the controller. It may draw items itself before, or after, further forwarding the request to the currently active representation.

When the state of an representation changed, e.g. because of user interaction, the representation may invoke the controller's `repaint()` method to trigger a fresh paint of the whole screen (which does simply a forward to the MIDlet's `repaint()` method).

FIFTH THIRD EX1062
PART 2 of 3
Page 237 of 379

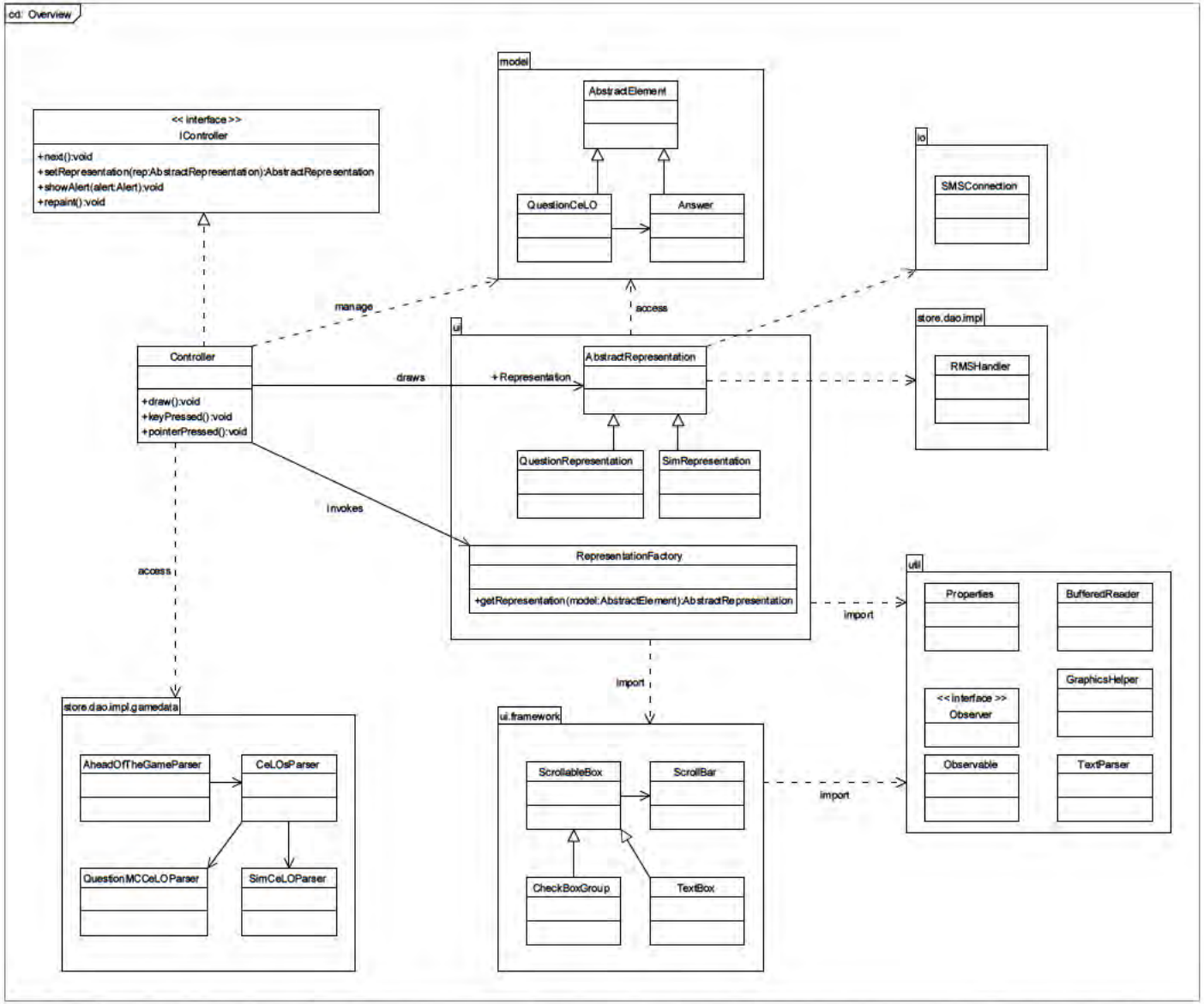


Figure 6.4: *Ahead of the Game* design overview (showing the most important representatives per package, only)

FIFTH THIRD EX1062
PART 2 of 3
Page 238 of 379

6.4.2 Threading

The usage of threads is limited to some occasions, e.g. the content parsing and simultaneously updating of the progress bar. In this example, also the observer pattern has been applied to manage updates.

While dealing with thread in the context of MIDlets, it is essential not to block the main thread (the one, the application is running in), because this would block the entire application and even key-strokes, repaints, or other events could not be processed — the whole application would hang.

6.4.3 XML-processing

Entry point for content parsing is always the `AheadOfTheGameParser` (in the left-bottom corner of figure 6.4). It instantiates the XML parser, reads some basic information (e.g. the joker definitions), and further forwards the parser instance to the `CeLOsParser`. The instance forwarding repeats several times, because the parser is divided into several pieces that are hierarchically organized.

If the `CeLOsParser` can't recognize the type of a CeLO element, the current XML element (and each sub-element) is ignored and an error message is printed on `stdout`.

Each parser requires a specific layout of XML elements, sub-elements and attributes, and may throw an exception if it is not met.

The used XML library is a minimal version of the freely available kXML 2 library[2]. The library is based on the Pull Parser API[3], which defines a new approach of reading XML documents, complementing *Document Object Model* (DOM) and *Simple API for XML* (SAX).

6.4.4 UI-Framework

Visual elements from the UI framework (see bottom of figure 6.4) are used to build up a representation. Standard J2ME libraries do not sufficiently support the required user interface design of the game, and even can't be extended to. Therefore the self made elements of this package build the basis of the whole application. The most important elements are:

- Scrollbar
- Textbox
- Checkbox-Group
- Radiobutton-Group

6.4.5 Representations

A representation combines several framework elements to one displayable screen (find some examples in the centre of figure 6.4). It manages the drawing of all components in the right order, as well as key and pointer events. Every representation is a subclass of `AbstractRepresentation`, which has basic implementations of the above described functionalities. Its main responsibilities can be classified in these categories:

[2]See: `http://kxml.sourceforge.net/`, accessed at 15.11.2007
[3]See: `http://xmlpull.org/`, accessed at 15.11.2007

FIFTH THIRD EX1062
PART 2 of 3
Page 239 of 379

- Background — e.g. a background image
- Foreground — e.g. a textbox that is shown in front of a background image
- Keystrokes- and touch-screen-events that are used to control the game. Each representation is responsible for handling it's own key- and touch-events.

6.4.6 Connection to the world

Data transfer to and from the game server comes in two flavours: SMS and HTTP. The user can choose his/her favourite protocol via the game settings (HTTP is faster, but in many countries still more expensive than SMS). The package including the communication classes is shown in the top-right corner of figure 6.4.

6.4.7 User Profiles

User profiles are stored in the device's RMS during the game. This is done for mainly two purposes: to access data (points, etc.) of previous mastered game levels (e.g. to calculate an overall statistic with the self-assessment element), and to transfer (overall) scoring information to the game server.

6.4.8 Utils

`MathHelper`, `GraphicsHelper`, classes to implement the observer pattern, and some others build a collection of useful tools. They are mostly used from within representation classes.

Properties

Configuration parameters of the whole application are centralized within this class (the Properties, as well as other utility classes are illustrated in the bottom-right corner in figure 6.4). For a later release version it's planned to outsource most of the properties into two distinct files: a configuration and a language file. The configuration includes e.g. the debug flag, parameters to configure the scoring system, and dimensions of bars, boxes, etc. The language file will contain all text snippets used by the game to create texts. It should easily be possible to switch from the standard language to another, by just replacing this one file.

6.5 Datamodel

figure 6.5 gives a good overview, of the different CeLO types and their interaction with each other. It also gives hints about how the application can be expanded, when needed.

The central element is the abstract class `CeLO` (shown in the centre of figure 6.5), where all other CeLO types are derived from it. Every CeLO may have one or more preparation CeLOs of any type, that are presented before the associated CeLO in sequential order. As every CeLO may have preparations, they may occur even on different levels (preparations of a preparation), although this is often not useful. However, the application does not count points that may result from preparation elements, irrespective of their level.

FIFTH THIRD EX1062
PART 2 of 3
Page 240 of 379

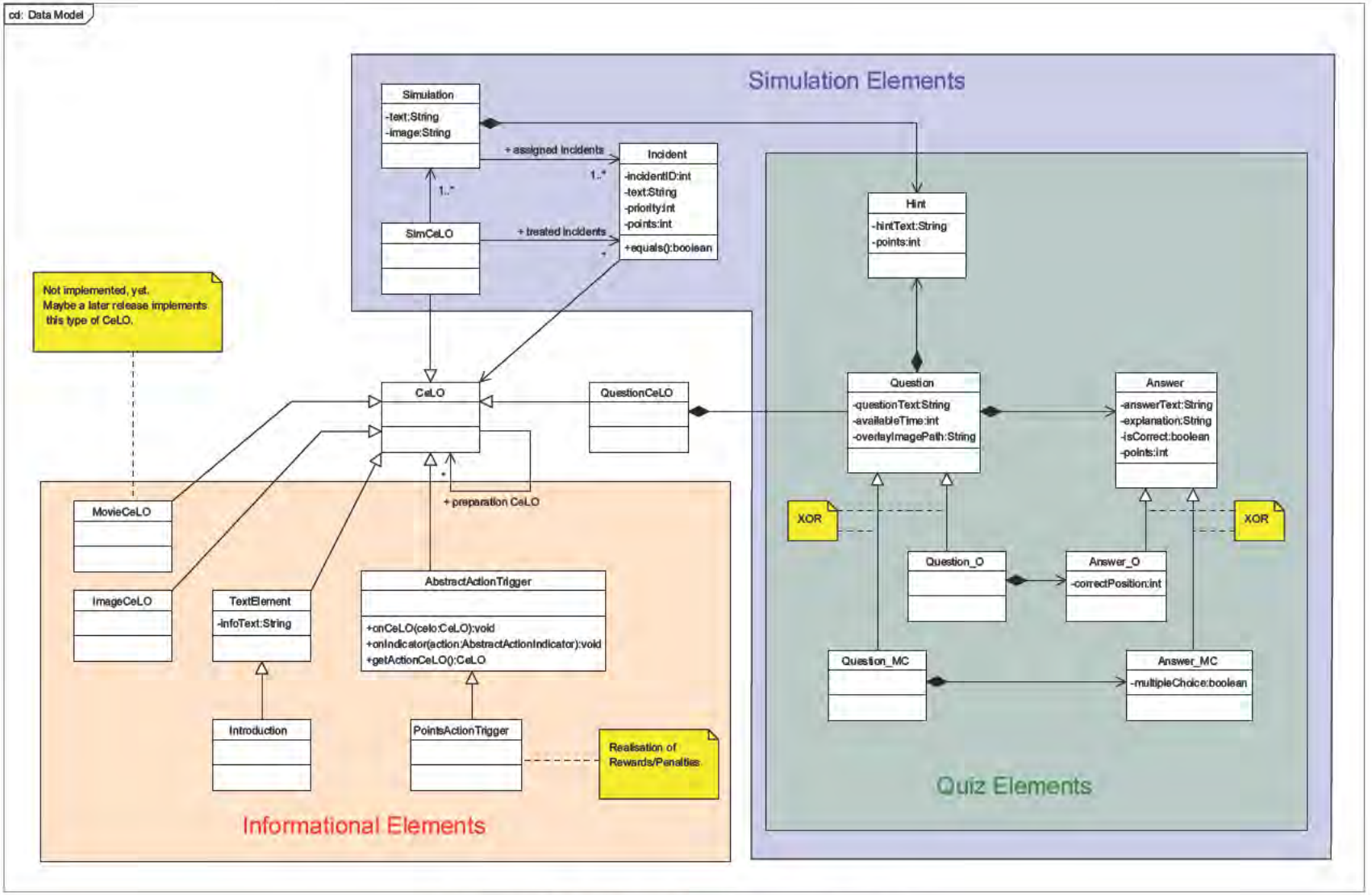


Figure 6.5: *Ahead of the Game* data model

FIFTH THIRD EX1062
PART 2 of 3
Page 241 of 379

Informational CeLOs are arranged in the bottom-left corner of figure 6.5. They have, in contrast to quiz or simulation elements, only informational character, what means that they do not interact with the user but rather present some kind of information.

On the right side of figure 6.5, all classes that are relevant for Question CeLOs are grouped together. To avoid co-variant problems, different subtypes of `Question` and `Answer` have been introduced.

Simulation elements are grouped in the upper area of figure 6.5, where the association from `Incident` to `CeLO` makes clear, that every CeLO can be used as incident-description. In contrast, the frame that claims the simulation area conform to the, in section 5.3 documented, specification of assigned questions, only. The difference of the specification and the more powerful implementation was the intention of those two visualisations.

6.5.1 Game Information

The following information can be authored for every game component:

- Time based point-calculation (yes|no)
- Permute Answers (show answers in a different order than authored)
- Number of available jokers for:
  - Hint
  - 50/50
  - Pause

6.5.2 General Information

A CeLO is the super-type for all learning objects and consists of the following data, which is inherited by all subtypes:

- Unique ID
- Corresponding Game-Level-ID

6.5.3 Quiz

A *Q-CeLO* is a subtype of a CeLO and adds the following data:

- Question
  - Unique Question ID
  - Text
- Answers, including for every answer:
  - Text
  - Correct = yes | no
  - Achievable points (can be negative)

FIFTH THIRD EX1062
PART 2 of 3
Page 242 of 379

- Short in-game explanation, why the model answer is correct[4].
- Hint + points (positive/negative) for using it
- Available time
- Overlay image

In accordance with the answer models employed, we define two subtypes of a Q-CeLO.

Q-CeLO-MK stands for two sub-classes of CeLOs, that are distinguished in the way, the user can select answer options. Therefore, this type extends a Q-CeLO with a Boolean value, which is used to distinguish between the two possibilities:

- Single-click (SK): the user picks one answer-option at a time — only one answer-option can be correct.
- Multiple-click (MK): one or more answer-options together form a correct response. Therefore, the author specifies one or more correct answer-options, the user has to match to gain all achievable points.

Q-CeLO-O the user has to bring the permuted answers in the correct ordering, what makes it essential to let the author define this order, and transfer this information to the client software. Therefore, an additional field is introduced, that stores the correct order for each answer-option.

6.5.4 Simulation (SimCeLO)

Each *SimCeLO* consists of typically four *Simulation*s, representing one "critical situation" as it develops over four points in time. For each simulation the author describes the situation at the current point in time, including a set of four *Incidents*, which also may have changed (as shown in figure 6.5):

- Simulation text
- Overlay image (optional)
- Hint (optional)
- A full set of Incidents, for each Incident must be authored:
  - Incident-ID
  - Unique-ID[5]
  - Priority
  - Points
  - A CeLO instance of any type (as shown in figure 6.5). Typically, a Question with the following characteristics is used for that purpose:

[4] It is possible to author additionally detailed end-game explanations, which is usually presented on the web-platform. Porting this feature to the client software could be part of future development.

[5] While the Unique-ID explicitly identifies one marked Incident, it's Incident-ID identifies a class of incidents, that represent the development of one incident over time.

FIFTH THIRD EX1062
PART 2 of 3
Page 243 of 379

* Question text
* Question image
* 2 answer options
* Hint
* Time (in seconds)
* etc.[6]

Typically, four incidents together form one situation, but the implementation allows n (where $n > 0$) incidents, which are different for n points in time. Please mind, that n deterioration levels requires a $n \times n$ matrix to be authored, which is a reasonable authoring effort, why we suggest that $n = 4$.

Example

Please find authoring examples in subsection 5.3.6 and appendix C.

6.5.5 Feedback

The user may answer a customisable question with free-text, which is sent to a special area on the server, where it's viewable by the author. The player may also cancel this dialogue in the case s/he has nothing to say, or does not want to pay for the data transfer. In this case, it's up to the teacher to interpret the missing notion.

6.5.6 Self-Assessment

The self assessment can be configured by changing the following parameters:

- Self-Assessment Text (e.g. "How do you think you have done?")
- n times (usually $n = 4$):
  - Threshold (e.g. 0.7, what means 70%)
  - Threshold-text (e.g. "good" or "excellent")

Several threshold and according text values create a classification, the user must use to categorize him/herself. The system is implemented very flexible, to make it adaptable to locale standards (for example US American use a different grading system than German schools).

6.5.7 Information (InfoCeLO)

This kind of representation consists of a background image and user specified text (e.g. for introduction purposes). The several subtypes differ in the instance of background image they use and where the text is positioned on the screen. Subtypes are:

- Info (background: flipchart)
- Introduction (background: boss)
- Reward / Penalty (background: various depending on reward/penalty)

[6] For further details see subsection 6.5.3.

FIFTH THIRD EX1062
PART 2 of 3
Page 244 of 379

6.5.8 System generated Information/Representations

There exist a few representations that cannot be customized by the author, but are generated by the application. These are:

- Gamelogo
- Startup-Screen
- Loading-Screen
- Highscores-Screen

6.6 Settings, Configuration

Settings configure the application. This can be done at several levels, but always has to be supported by the system and therefore specified before the implementation takes place. Most important settings are those configurable by users (authors and players), because they require an intuitive (graphical) user interface.

The settings listed below are divided into categories presenting roles of the people who are responsible for changing them (according to the above mentioned levels).

6.6.1 Player

The player can change the following game settings via the game's user interface:

- Sounds (on, off)
- Server-connection (SMS vs. HTTP, incl. time-outs and retries)
- Server-address (could be provided during deployment automatically, but can be changed by the user)
- user-name and password

6.6.2 Author

Via the web-platform, the author can change the following data:

- Jokers
- Meta-information (title, topic, sub-topic, etc.)
- Content + Timing + Scoring
- Security (allowed users/user-groups)

FIFTH THIRD EX1062
PART 2 of 3
Page 245 of 379

6.6.3 Developer

Developers make a pre-configuration of the game, by changing the following settings during the implementation phase:

- Activate debug mode
- System generated text-snippets
- File-names (e.g. config-file)
- Image-paths
- Minimum and maximum supported screen resolutions and Java APIs (an error message will be generated if the resources don't match the requirements)
- Colours, sizes, icons, images, ...

6.7 Server Communication

As described in subsection 5.1.7, it was planned to add some competition with other users to the game. To reduce costs, a communication flow between server and client, visualised in figure 6.6, has been developed. Because the mentioned competition is not implemented yet, the diagram could be reduced to the communication-flow at game end. However, for future work, the whole planned communication flow is described in the next sections, where it should be obvious which of them are not relevant for the current implementation.

6.7.1 Distribution

First of all the user needs to download the game to his/her mobile phone (described on page 23), which is the first interaction with the *mGBL*-server.

6.7.2 Game Start

At game start, the application sends a HELLO message to the server to trigger the actual highscores to be sent.

The server responds to the HELLO message with the current highscores for all non-finished levels as well as the avatar id (i.e. the bosses sex) of the users in that list.

6.7.3 Game end

At game end, the current user profile[7] will be transferred to the server, what requires the user's authorization.

The actual ranking will be calculated on the server and transferred to the client, so it only needs to present it on the screen (without doing any calculations).

[7] For detailed information concerning the user profile, please see section 6.10.

FIFTH THIRD EX1062
PART 2 of 3
Page 246 of 379

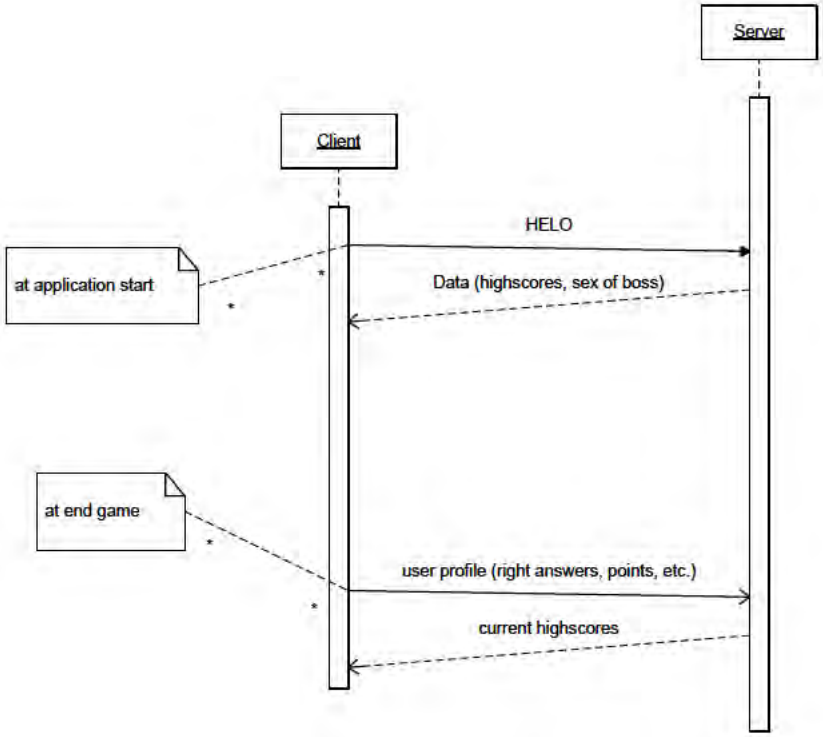


Figure 6.6: Schematic client-server communication

FIFTH THIRD EX1062
PART 2 of 3
Page 247 of 379

6.7.4 Feedback[8]

As for all other server communications (e.g. transfer of free-text answers, see subsection 6.5.5), the user specifies (via the settings dialogue) whether to use SMS or HTTP based connections. The user composed feedback is sent immediately to the server.

6.7.5 Component-Synchronization

When playing a hybrid game composed of the quiz and the simulation component, the simulation must not be played before the quiz has been successfully completed. To ensure the predefined order, the server must ensure that the second component is not downloadable before the first has been finished.

6.8 File Structure

The game-data file stores content concerning all CeLO types of the game. Although differently specified[9], it's possible to mix Simulation-CeLOs and other types of CeLOs within one game as well one level. However, this should not be considered as a drawback, but as a feature — the application allows content structures like specified, and even more complex ones.

The most important characteristics of the content file are:

- Levels are numbered consequently - there is no missing level.
- Level 0 indicates a general introduction.
- There must be at least one answer option for each question (independently of the question's type)[10].
- Optional parameters may be omitted.

The exact definition of the content file, specified in XML's *Document Type Definition* (DTD), can be found in the appendix (listing A.1).

6.9 Highscores

At game end, the ranking will be presented to the user. The presentation is intelligent in the way, that the user sees e.g. not only the top 5, but always his/her own rank, beside some key-ranks. Let's, say for example, the user becomes 17^{th}, s/he will see the top three, his/her own rank, and the last place.

It is planned to implement the possibility to keep the user updated even after game end, so that s/he can subsequently ask for his/her ranking via a game's menu entry. In either case, users have access to their scores via the web-platform.

[8] For further information, please see subsections 5.2.3 and 6.5.5

[9] See chapter 5

[10] There should be two or more answer options, otherwise multiple-choice makes no sense.

FIFTH THIRD EX1062
PART 2 of 3
Page 248 of 379

6.10 User Profiles

The information stored in the profile and sent to the server is:

- User-ID (indirectly by MSISDN, or via user-name & password credentials)
- Overall achieved points

Advanced versions of the game may transfer more detailed statistical information, such as on a per-CeLO basis:

- CeLO-ID
- Selected answer-options (for Q-CeLOs)
- Correctness of answers
- Used jokers
- Used time

6.11 Game Deployment

The deployment tool will be seamlessly integrated into the server platform, and is responsible for creating the downloadable jar-files out of the authored data and the game template for different target devices. Schematic representations of the game creation process, in different views, are shown in figures 4.1, 6.1 and 6.2.

FIFTH THIRD EX1062
PART 2 of 3
Page 249 of 379

Chapter 7

Evaluation

7.1 Hype or Tripe

Georg Ströhlein tries to identify characteristics of mobile learning scenarios that identify them whether as hype or tripe [Str07]. He admits that mobile learning has the potential of being a hype, when it is ready for everyday's usage by relying on widely spread technology, or a tripe, when it can not. The consent of his work was, that it heavily depends on the realization of the learning process or model, whether it is the one or the other. Based on continuous user trials, we tried to identify and eliminate weaknesses of *Ahead of the Game*. Overall we can now say that our game template is on the hype-side, although there are some aspects that need to be improved in future versions. Therefore, user's concerns are documented in the next chapter, while conclusions of these outcomes as well as ideas for improvements that don't come from user trials are summed up in section 7.3.

7.2 Feedback from User Trials

User trials have been conducted at different formal levels and in different states of development to address different questions.

First of all some kind of slide-show with static mock-ups has been implemented for mobile phones. These mock-ups were pictures presenting the basic elements of the user interface to get information about dimensions, colours, and arrangement of elements on a device's display.

An early implementation of the game was then developed, mainly to work on PC emulators for the 1^{st} *user trials*. It has been presented to many potential users of both targeted user groups – students and teachers – in several countries to get an impression of how they would accept the game idea as well as the user interface. About 150 participants (13 teacher, the rest students) attended the 1^{st} user trials online, where the majority liked the presented concept of *Ahead of the Game*.

After an elaborate enhancement of the learning model[1] and its implementation, a small, unofficial user trial was conducted to check the clarity and comprehensibility of the new game concept. This survey has been conducted at the *Microlearning Conference*, which was considered appropriate and meaningful to test a mobile game. The very useful hints for improvements could be found on three levels:

[1] The quiz component was enhanced and set into a common context with the newly established simulation component.

FIFTH THIRD EX1062
PART 2 of 3
Page 250 of 379

- User interface and its design elements: the font size was considered too small, while the separation of question and answer-options into two distinct boxes was confusing the participants.
- Storyline and appropriateness of system responses: participants were irritated because of some similar questions and simulation-situations of the game[2]. We learned from these responses that the preparation of the content and its sequence is at least as important as the user interface or the learning theories that are the basis of the game.
- Interaction of software and hardware: too long loading times, and, in contrast, too fast responses on selecting or deselecting answer options via the touch screen have been reported, as well as too long responses when clicking the *next* button. Unfortunately, this is the level we have the least impact on, because it mainly concerns the underlying hardware, which is sometimes quicker or slower, depending on the used device.

2^{nd} *user trials* have, again, been conducted within the context of the *mGBL* project. Content, in the field of e-commerce, has been newly created and translated into the language of the hosting country (Slovenia). Participants where 6 teachers and 5 students, where all but one have been provided with mobile phones. One teacher brought her own cell phone, where the game has been installed onto. Results of the these user trials showed that the user interface could be further improved (or reconsidered). Because there were hardly negative comments regarding the content or its storyline and the hardware-software interaction (see above), we conclude that these issues have been resolved. Instead, we received a lot of feature requests, where the most prominent were the support of video-clips and a multi-player mode.

The user interface It has been reported that not enough content is shown at a time and the scoring was not successfully explained (especially that of multiple-answer questions), where the latter point was mainly raised by students. A discrepancy between teachers and students was found on comments concerning the controls of the game. While students were satisfied with the controls, although they needed some time to get familiar with the given device, teachers found that controls were not clear enough. Although, teachers did not pay much attention on the game design, they, as well as students, recommended to add animations as some kind of variation.

Content and storyline were considered properly, although some students mentioned they would not pay any attention if texts were longer, but select an answer randomly.

The simulation was flavoured by teachers, who thought it is a good way to make people familiar with real life situations in a simulated, thus safe, environment. They also had the idea to support some kind of decision trees, where the decision of one situation puts the player in another one that s/he has to cope with again[3].

[2]Indeed, the content was not prepared carefully enough, so that it seemed to some participants to have a déjà vu.

[3]For example, the user is put in the situation to help one of four people that are hurt in a car accident, s/he picks one and finds him/herself in a situation where to cope with four injuries (e.g.: head wound, broken leg, swollen eye, stomach-ache) of that particular person.

FIFTH THIRD EX1062
PART 2 of 3
Page 251 of 379

Other issues concerned mainly multimedia content (video clips) and a multi-player version of the game. Students also demanded some kind of statistical analysis tool to help teachers identify possible weaknesses of particular students, student groups or the game content.

The male boss was considered inappropriate by most participants in several user trials. Therefore we added the possibility to switch, at the beginning of the game, to a female boss character. Although this feature was implemented in an early stage of development, people still reported that they didn't like the picture. It seems that it is really essential to replace this character with a more appropriate one.

7.3 Ideas for Improvements

The game conception phase and the user trials produced some ideas for improvements and nice-to-haves. To sketch the potential of the implemented prototype, these are listed below, starting with important improvements, followed by more exceptional ideas.

A better explanation of points is essential before making the game available for the broad public. User trials showed (see section 7.2), that it is not clear enough to users, why they get a certain amount of points. Because additional plus or minus points may be gained by using a joker, these circumstances must also be explained sufficiently.

The scoring model must be improved to make it more acceptable for students. User trials showed that they are not comfortable with getting 0 points for a partially right multiple-answer question. Therefore, the scoring model of this CeLO type needs to be reconfigured in a way that partially right answers bring points, too. On the one hand, this may seduce some students to cheat — checking all answer options, a random selection, or a certain pattern (e.g., options 1, 3 and 4) may lead to reduced, but still positive scores. On the other hand, a reworked scoring model combined with accurate authoring may reduce cheating to a minimum.

Animations were considered as both fun and distraction, within our 2^{nd} user trials. Because of this divergent meanings, implemented animations have to be both discreet and appealing at the same time. However, animations are, as all of us know, an integral part of all games, because they entertain and attract, and are indispensable, therefore.

Statistical analysis of the participants' progress and its availability on the server platform for the student and his/her teacher. Tracking each step the player makes and transferring this statistical data to the server is a good possibility for teachers to determine weaknesses of individual players, or the whole group. Instead of finding weaknesses in the group of students, such a tool could be used to make weaknesses in the authored content or the whole curricular visible.

Reveal hidden features or bonus material by extraordinary good or bad scores, interesting (but not necessarily correct) answers, or by a system generated random event adds some value of uncertainty and engages players to keep playing and reach (again) good scores [SZ03]. Examples

FIFTH THIRD EX1062
PART 2 of 3
Page 252 of 379

are medals for outstanding scores, that are visible to all players via the web-platform, or additional CeLOs that may bring bonus points or free hints for questions to come.

A Multi-player mode can be implemented in a direct or indirect way. The indirect way makes the game round based, where one game level correlates to one round. At the beginning of each round, the client software requests the current highscores from the server and presents the leading person as the player's boss who is challenging him/her. Additionally the complete highscore list (including the player's score) may be shown at the end of each round. This kind of multi-player mode was initially planned, but not implemented, because of time constraints due to changing requirements (see subsection 5.1.7). A direct multi-player mode is much more complicated, without negating the advantages of time and space independence, that people love so much about mobile devices. One possible approach is to make one player, who has finished the game already, a validator of one or more others. S/he receives the player's challenge and his/her solution and has to judge the solution as right or wrong. A correct judgement brings the validator bonus points, while the player earns additional points for a correct solution with an incorrect judgement. However, the group of participating people must be large enough to ensure a proper validation service, and thus a continuous game flow.

FIFTH THIRD EX1062
PART 2 of 3
Page 253 of 379

Chapter 8

Conclusion

In this work I have presented basics, and especially benefits of mobile learning (section 2.3), game based learning (section 2.2), and the combination of both areas (section 2.4). While the main benefits of game based learning is to be found in the learner's motivation and engagement with the topic, mobile learning enables students to follow their own preferences in learning while keeping focused for a longer period.

Based on state of the art development technology (*JavaME*), this work focuses on the specification and implementation of a prototypical mobile learning game for mobile phones. The whole architecture is object and performance oriented, to address both, maintainable source code, and the user's comfort of fast system responses, respectively.

User tests showed that we (participants of the EU project *Mobile Game-Based Learning*) are on the right track with our implementation, but it is a long road to go, until an attractive, engaging and long lasting mobile learning game will result from our efforts. Particularly, a game *template* is even more laborious to develop, because it has not only to be attractive and engaging, but must be applicable to a broad spectrum of different, often orthogonal, contents. The list of requirements does not end at the client application, but requires an equally high qualitative authoring tool to get the content together with the template. The tool has to be balanced between an intuitive user interface and the power of a programming language, tight restrictions to enforce high quality content and the possibility to follow own ideas.

Concluding, I believe, technology will become even better and in a few years from now mobile phones will have the power of a today's game console, not only talking of its processing power but also of the provided development tools. Therefore, we have to continue working now, to find ways to motivate learners of any age, to have mature learning models ready, when technology is.

FIFTH THIRD EX1062
PART 2 of 3
Page 254 of 379

Chapter 9

Bibliography

[Abt87] Clark C. Abt. *Serious Games*. University Press of America, Mai 1987.

[ASS71] Elliot M. Avedon and Brian Sutton-Smith. *The Study of Games*. John Wiley, New York, 1971.

[Att04] Jill Attewell. Mobile technologies and learning. Technical report, Technology Enhanced Learning Research Centre, 2004.

[Com] Java Community. Java specification requests. `http://www.jcp.org/en/jsr/all`, accessed at 10.10.2007.

[DRL96] John V. Dempsey, Karen Rasmussen, and Barbara Lucassen. The instructional gaming literature: Implications and 99 sources. Technical Report 96-1, University of South Alabama, 1996.

[Ess06] Sebastian Esser. *Inside Mobiles – Software für Mobiltelefone*. Vdm Verlag Dr. Müller, 2006.

[Hla06] Peter Hlavac. Innovative user interfaces for accessing music on mobile devices. Master's thesis, TU Wien, 2006.

[HS07] Richard Harrison and Mark Shackman. *Symbian OS C++ for Mobile Phones*, volume 3. Wiley & Sons, June 2007.

[JPa] J2ME-Polish. J2ME-Polish name clarification. `http://www.j2mepolish.org/faq.html#name`, accessed at 10.10.2007.

[JPb] J2ME-Polish. Nokia 6230i memory limitations. `http://www.j2mepolish.org/devices/Nokia/6230i.html#memory`, accessed at 10.10.2007.

[LJD07] Richard Leggett, Scott Janousek, and Weyert DeBoer. *Foundation Flash Applications for Mobile Devices*. Computer Bookshops, January 2007.

[Mal81a] Thomas W. Malone. Toward a theory of intrinsically motivating instruction. *Cognitive Science*, 5(4):333–369, 1981.

[Mal81b] Thomas W. Malone. What makes computer games fun? *Byte*, 6:258–277, December 1981.

[Mat] School Matters. Mobile phones mobile minds. Available at: `http://video.google.com/videoplay?docid=1043898959196049305`, accessed at 1.11.2007.

FIFTH THIRD EX1062
PART 2 of 3
Page 255 of 379

[PKD04] Maja Pivec, Anni Koubek, and Claudio Dondi. *Guidelines for Game-Based Learning.* Pabst Science Publishers, 2004.

[Pre01] Marc Prensky. *Digital Game-Based Learning.* McGraw-Hill Education, 2001.

[Rod04] Leon Rodin. E-learning to m-learning: an investigation into the potential for content conversion. In Jill Attewell and Carol Savill-Smith, editors, *Mobile learning anytime everywhere*, pages 171–176, 2004. Available at: `http://www.m-learning.org/archive/docs/MLEARN%202004%20book%20of%20conference%20papers.pdf`, accessed at 10.10.2007.

[Sch07] Klaus-Dieter Schmatz. *Java Micro Edition.* Dpunkt Verlag, 2. edition, 2007.

[SHB04] Liina Stotz, Gabriela Hoppe, and Prof. Dr. Michael H. Breitner. Interaktives Mobile(M)-Learning auf kleinen Endgeräten wie PDAs und Smartphones. IWI Discussion Paper Series 12, Institut für Wirtschaftsinformatik, Universität Hannover, August 2004. Available at: `http://ideas.repec.org/p/ifw/iwidps/iwidps12.html`, accessed at 10.10.2007.

[Str07] Georg Ströhlein. Mobile learning using mobiles: Hype or tripe? In *Neue Trends im E-Learning*, pages 1–15, Juni 2007.

[Sun01] Todd Sundsted. J2ME grows up – New specs signal a growth spurt for Java 2 Platform's smallest edition, May 2001. `http://www.ibm.com/developerworks/java/library/j-j2me/`, accessed at 15.11.2007.

[SZ03] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals.* The MIT Press, November 2003.

[TB04] John Traxler and Nell Bridges. Mobile learing – the ethical and legal challenges. In Jill Attewell and Carol Savill-Smith, editors, *Mobile learning anytime everywhere*, pages 203–207, 2004. Available at: `http://www.m-learning.org/archive/docs/MLEARN%202004%20book%20of%20conference%20papers.pdf`, accessed at 10.10.2007.

[Wika] Wikipedia. Adobe Flash Lite. `http://en.wikipedia.org/wiki/Flash_Lite`, accessed at 15.11.2007.

[Wikb] Wikipedia. JAR file format. `http://en.wikipedia.org/wiki/JAR_(file_format)`, accessed at 10.10.2007.

[Wikc] Wikipedia. Java community process. `http://en.wikipedia.org/wiki/Java_Community_Process`, accessed at 15.11.2007.

[Wikd] Wikipedia. Java version history. `http://en.wikipedia.org/wiki/Java_version_history`, accessed at 10.10.2007.

[Wike] Wikipedia. Java virtual machine: Bytecode verifier. `http://en.wikipedia.org/wiki/Java_Virtual_Machine#Bytecode_verifier`, accessed at 10.10.2007.

[Wikf] Wikipedia. MIDlet. `http://en.wikipedia.org/wiki/Midlet`, accessed at 10.10.2007.

FIFTH THIRD EX1062
PART 2 of 3
Page 256 of 379

[Wikg] Wikipedia. Mobile learning. `http://en.wikipedia.org/wiki/Mobile_learning`, accessed at 1.11.2007.

[Wikh] Wikipedia. .Net compact framework. `http://en.wikipedia.org/wiki/.NET_Compact_Framework`, accessed at 15.11.2007.

[Wiki] Wikipedia. Symbian os. `http://en.wikipedia.org/wiki/Symbian_os`, accessed at 15.11.2007.

[Wikj] Wikipedia. Unreal game engine. `http://de.wikipedia.org/wiki/Unreal_Engine`, accessed at 01.11.2007.

[Wikk] Wiktionary. Game. `http://en.wiktionary.org/wiki/game`, accessed at 01.11.2007.

[Wor] WordNet. Wordnet 2.1 search-term 'game'. Available at: `http://wordnet.princeton.edu/obtain`, accessed at 01.11.2007.

[YD04] Paul Yao and David Durant. *.Net Compact Framework Programming with C (C Sharp)*. Addison-Wesley Longman, May 2004.

[Zaca] Erik Zachte. Wikipedia access statistics. `http://stats.wikimedia.org/EN/TablesUsagePageRequest.htm`, accessed at 10.10.2007.

[Zacb] Erik Zachte. Wikipedia usage statistics. `http://stats.wikimedia.org/EN/ChartsWikipediaEN.htm`, accessed at 10.10.2007.

[ZHKH07] Weishan Zhang, Dong Han, Thomas Kunz, and Klaus Marius Hansen. Mobile game development: Object-orientation or not. In *Computer Software and Applications Conference, 2007. COMPSAC 2007 - Vol. 1. 31st Annual International*, pages 601–608, July 2007.

FIFTH THIRD EX1062
PART 2 of 3
Page 257 of 379

Appendix A

File Format

The format of the content file, specified in DTD-Syntax, can be found in listing A.1.

Listing A.1: The format of the content file, specified in DTD-Syntax

```
<!ELEMENT AheadOfTheGame (Jokers?, Level+)>
    <!ELEMENT Jokers (Joker+)>
        <!ELEMENT Joker EMPTY>
            <!ATTLIST Joker
                type CDATA #REQUIRED
                number CDATA #REQUIRED
            >

    <!ELEMENT Preperation (QuestionMCCeLO|QuestionOCeLO|SimCeLO|Info|
        Introduction|Image|ImageText|QuestionFreeText|SelfAssessment|
        Reward)+>

    <!ELEMENT Level (QuestionMCCeLO|QuestionOCeLO|SimCeLO|Info|
        Introduction|Image|ImageText|QuestionFreeText|SelfAssessment|
        Reward)+>
        <!ATTLIST Level
            level CDATA #REQUIRED
        >

        <!ELEMENT Reward (Preperation?, History, Action+)>
            <!ATTLIST Reward
                id              CDATA           #REQUIRED
            >

            <!ELEMENT History EMPTY>
                <!ATTLIST History
                    history         (all)       'all'
                >
<!--                    history         (all|level|last)    'all' -->

            <!ELEMENT Action (QuestionMCCeLO|QuestionOCeLO|SimCeLO|Info
                |Introduction|Image|ImageText|QuestionFreeText|
                SelfAssessment)>
                <!ATTLIST Action
                    comp            (lt|gt|eq)          'eq'
                    value           CDATA               #REQUIRED
                >
```


FIFTH THIRD EX1062
PART 2 of 3
Page 258 of 379

```
<!ELEMENT QuestionFreeText (Preperation?)>
    <!ATTLIST QuestionFreeText
        id              CDATA           #REQUIRED
        question        CDATA           #REQUIRED
        answer          CDATA           #IMPLIED
    >

<!ELEMENT SelfAssessment (Preperation?, SelfAssessmentThreshold
    +)>
    <!ATTLIST SelfAssessment
        id              CDATA           #REQUIRED
        question        CDATA           #REQUIRED
    >

<!ELEMENT SelfAssessmentThreshold EMPTY>
    <!ATTLIST SelfAssessmentThreshold
        threshold       CDATA           #REQUIRED
        text            CDATA           #REQUIRED
    >

<!ELEMENT Info (Preperation?)>
    <!ATTLIST Info
        id              CDATA           #REQUIRED
        text            CDATA           #REQUIRED
    >

<!ELEMENT Image (Preperation?)>
    <!ATTLIST Image
        id              CDATA           #REQUIRED
        picture         CDATA           #REQUIRED
    >

<!ELEMENT ImageText (Preperation?)>
    <!ATTLIST ImageText
        id              CDATA           #REQUIRED
        picture         CDATA           #REQUIRED
        text            CDATA           #REQUIRED
        overlay         (true|false)    'true'
    >

<!ELEMENT Introduction (Preperation?)>
    <!ATTLIST Introduction
        id              CDATA           #REQUIRED
        text            CDATA           #REQUIRED
    >

<!ELEMENT QuestionMCCeLO (Preperation?, AnswerMC+, Hint?)>
    <!ATTLIST QuestionMCCeLO
        id              CDATA           #REQUIRED
        question        CDATA           #REQUIRED
        multipleChoice  (true|false)    'false'
        picture         CDATA           #IMPLIED
```


FIFTH THIRD EX1062
PART 2 of 3
Page 259 of 379

```
            timeAvailable   CDATA           #REQUIRED
        >

        <!ELEMENT AnswerMC EMPTY>
            <!ATTLIST AnswerMC
                answer      CDATA           #REQUIRED
                explanation CDATA           #REQUIRED
                points      CDATA           #REQUIRED
                correct     (true|false)    'false'
            >

        <!ELEMENT Hint EMPTY>
            <!ATTLIST Hint
                hint    CDATA   #REQUIRED
                points  CDATA   #REQUIRED
            >

    <!ELEMENT QuestionOCeLO (Preperation?, AnswerO+, Hint?)>
        <!ATTLIST QuestionOCeLO
            id              CDATA           #REQUIRED
            question        CDATA           #REQUIRED
            picture         CDATA           #IMPLIED
            timeAvailable   CDATA           #REQUIRED
        >

        <!ELEMENT AnswerO EMPTY>
            <!ATTLIST AnswerO
                answer      CDATA           #REQUIRED
                explanation CDATA           #REQUIRED
                points      CDATA           #REQUIRED
                order       CDATA           #REQUIRED
            >

    <!ELEMENT SimCeLO (Preperation?, Simulation+)>
        <!ATTLIST SimCeLO
            id  CDATA #REQUIRED
        >

        <!ELEMENT Simulation (Incident+, Hint?)>
            <!ATTLIST Simulation
                id              CDATA           #REQUIRED
                question        CDATA           #REQUIRED
                picture         CDATA           #IMPLIED
            >

        <!ELEMENT Incident (QuestionMCCeLO|QuestionOCeLO|SimCeLO|
            Info|Introduction|Image|ImageText|QuestionFreeText|
            SelfAssessment|Reward)>
            <!ATTLIST Incident
                incident-id     CDATA           #REQUIRED
                id              CDATA           #REQUIRED
                text            CDATA           #REQUIRED
```


FIFTH THIRD EX1062
PART 2 of 3
Page 260 of 379

```
                    prio            CDATA           #REQUIRED
                    points          CDATA           #REQUIRED
                >
```

Appendix B

Game Examples

Figures B.1 and B.2 exemplary show the game-flow of a quiz and a simulation component, respectively. The first (figure B.1) presents the three different question types (Ordering, Single-Click and Multiple-Click) and the system-reaction on wrong and right user input.

A look at figure B.2, gives an impression of how a simulation, and especially deterioration, works. For clearness, only two of four possible incidents are presented here, while the descriptions of the different situations have been omitted for the same reason. At the two remaining incidents, you can see how they are deteriorated over time, where it might occur – like for the second incident ("unconscious man") – that a patient dies. From a complete 4×4 matrix of all incidents only one element per row is presented to the user, depending on the deterioration level.

FIFTH THIRD EX1062
PART 2 of 3
Page 262 of 379

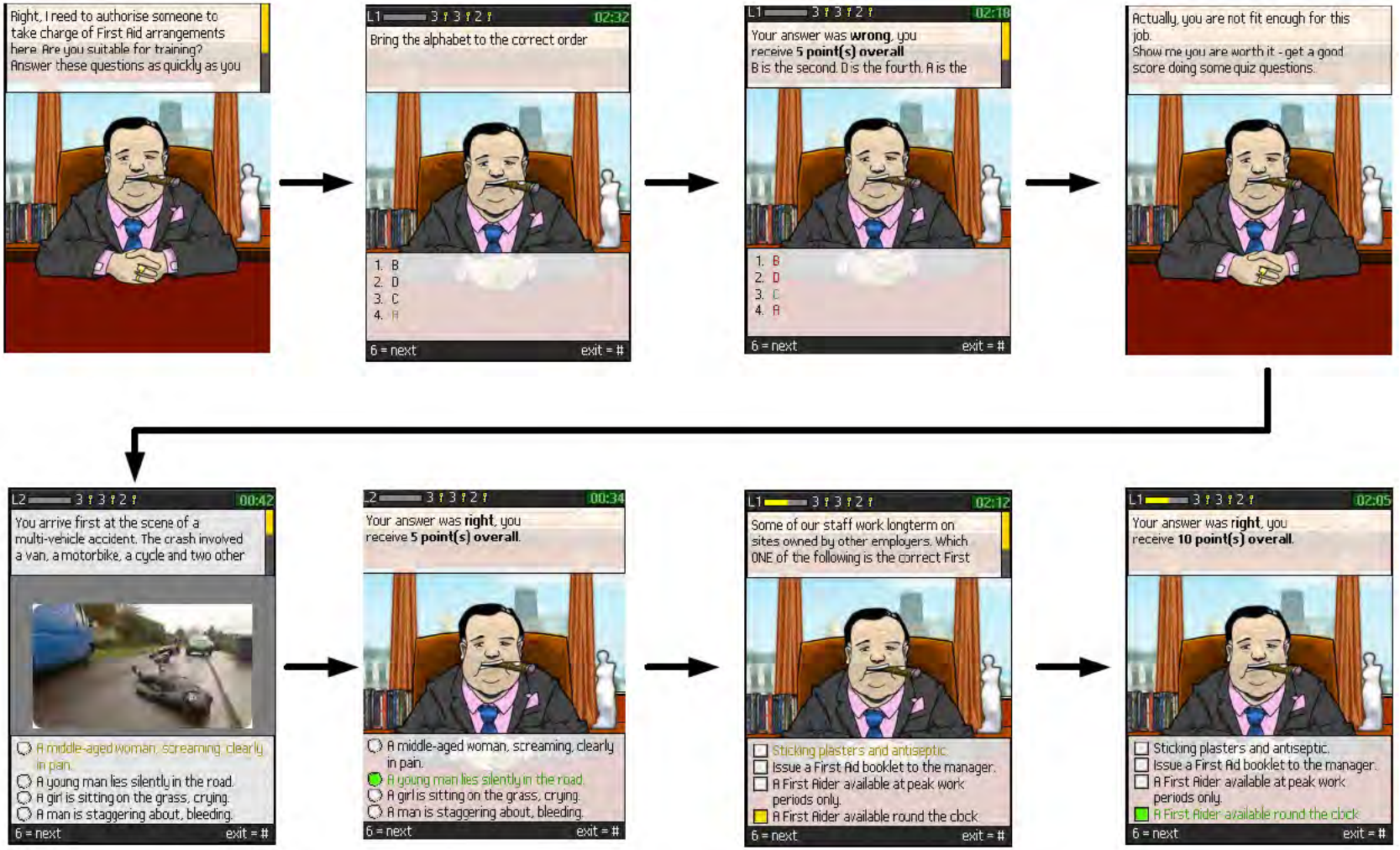


Figure B.1: Exemplary game-flow of a quiz component.

FIFTH THIRD EX1062
PART 2 of 3
Page 263 of 379

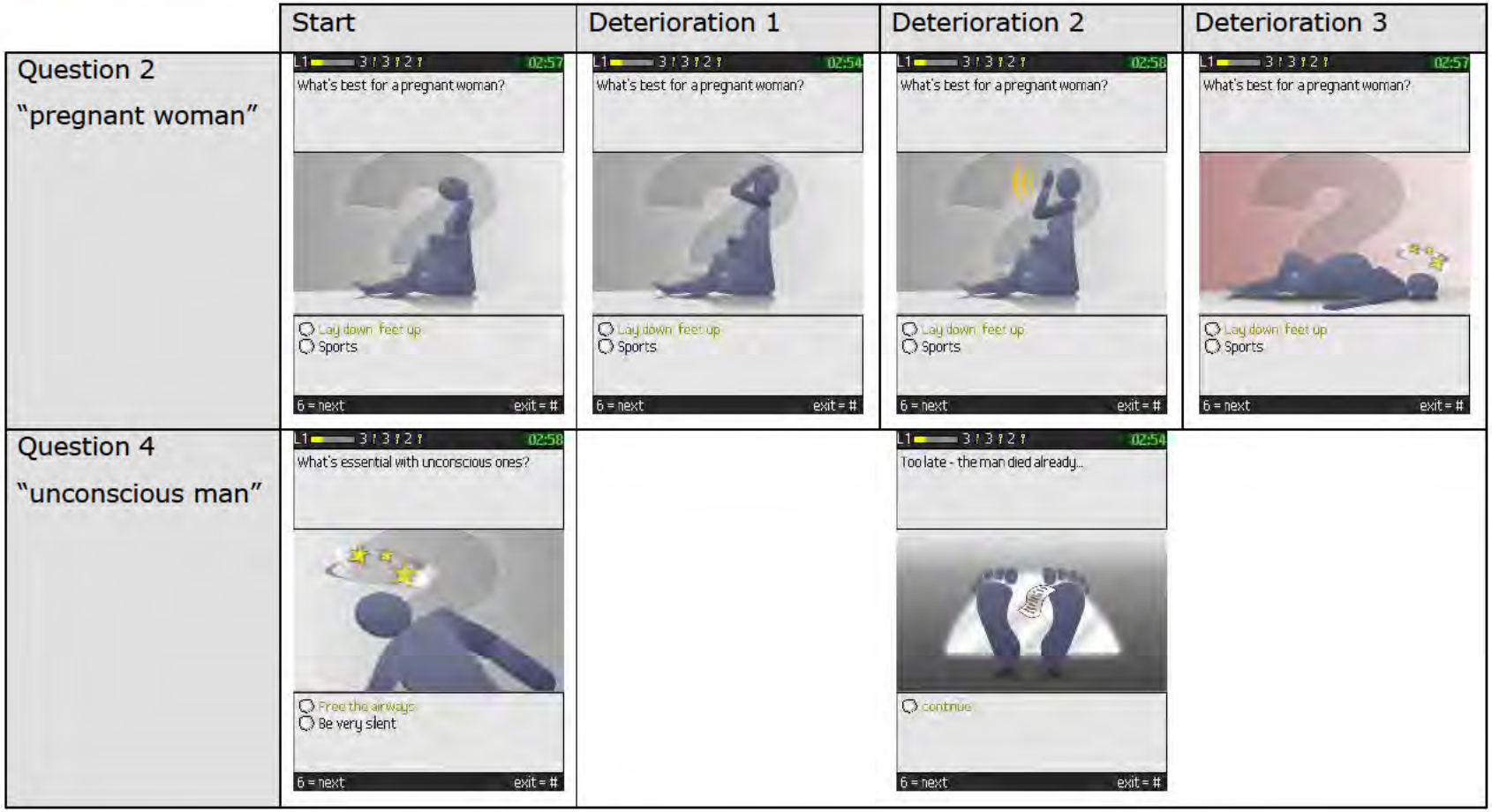


Figure B.2: Two incidents of a simulation component, and their deterioration over time.

FIFTH THIRD EX1062
PART 2 of 3
Page 264 of 379

Appendix C

Authoring Interface

To give an impression of how the authoring may work, I want to present some exemplary design mock-ups here. Figures C.1 and C.2 show the authoring interface, mainly used for the quiz component. The difference between both can be seen at the bottom of the graphics, where the answer-options are edited — the one uses radio-buttons, the other check-boxes. On the right side the author is provided with an overview, of the so far edited elements, while the edited game-levels are shown at the top. Usually, the authoring starts by choosing the card-type (SC, MC, Simulation, etc.), and editing the question text. Beside a custom image (left side of figure C.1), the author may specify a hint including bonus or penalty points for using it. The difficulty-level at the top and the radio-buttons below the hint, help unexperienced users at their first steps.

Figure C.3 presents the authoring design for the simulation component. As you can see, the navigational elements are the same as for the quiz component, while the content creation is a bit more complex. Therefore, it is possible to fold elements to a small, representative overview, to be able to keep an overview of the whole picture. The several deterioration levels are arranged from left to right (separated by dotted lines), while details of each level are arranged vertically. In the example (figure C.3), the first incident at the first deterioration level has been already edited. To make the user interface look not too overcrowded, an incident can have a Single-Click Question with exactly two answer-options, only.

FIFTH THIRD EX1062
PART 2 of 3
Page 265 of 379

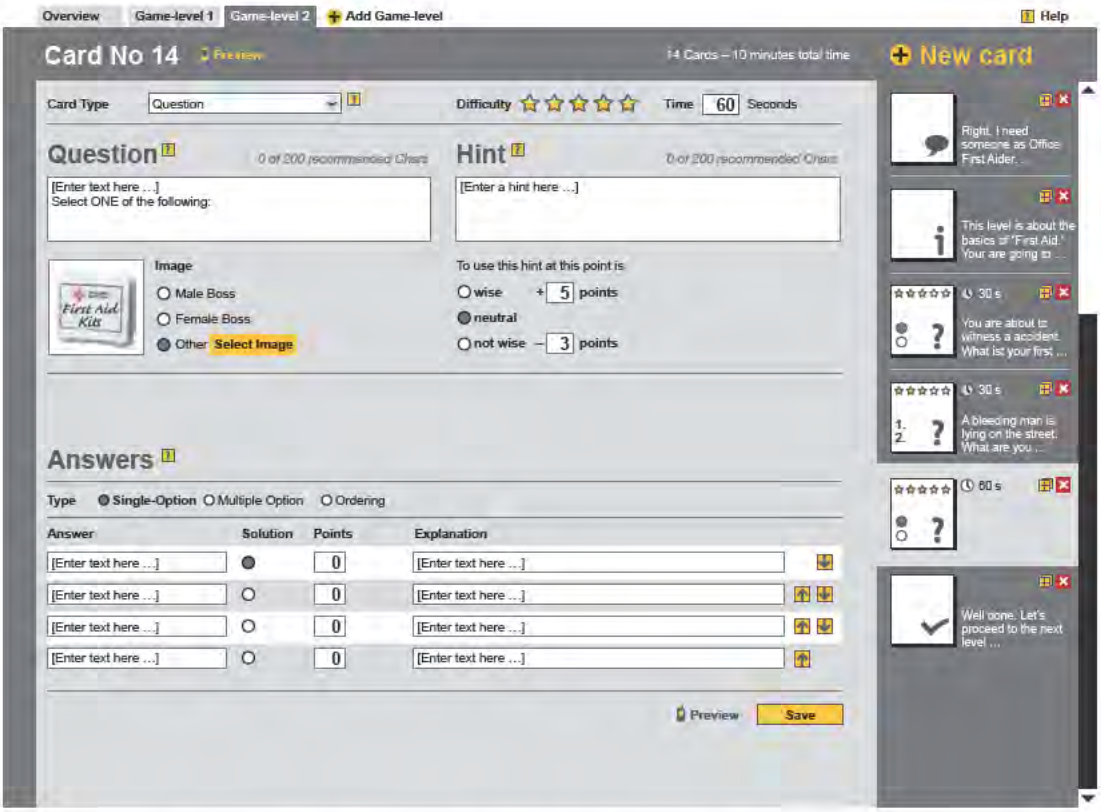


Figure C.1: Authoring card for a Single-Click Question.

FIFTH THIRD EX1062
PART 2 of 3
Page 266 of 379

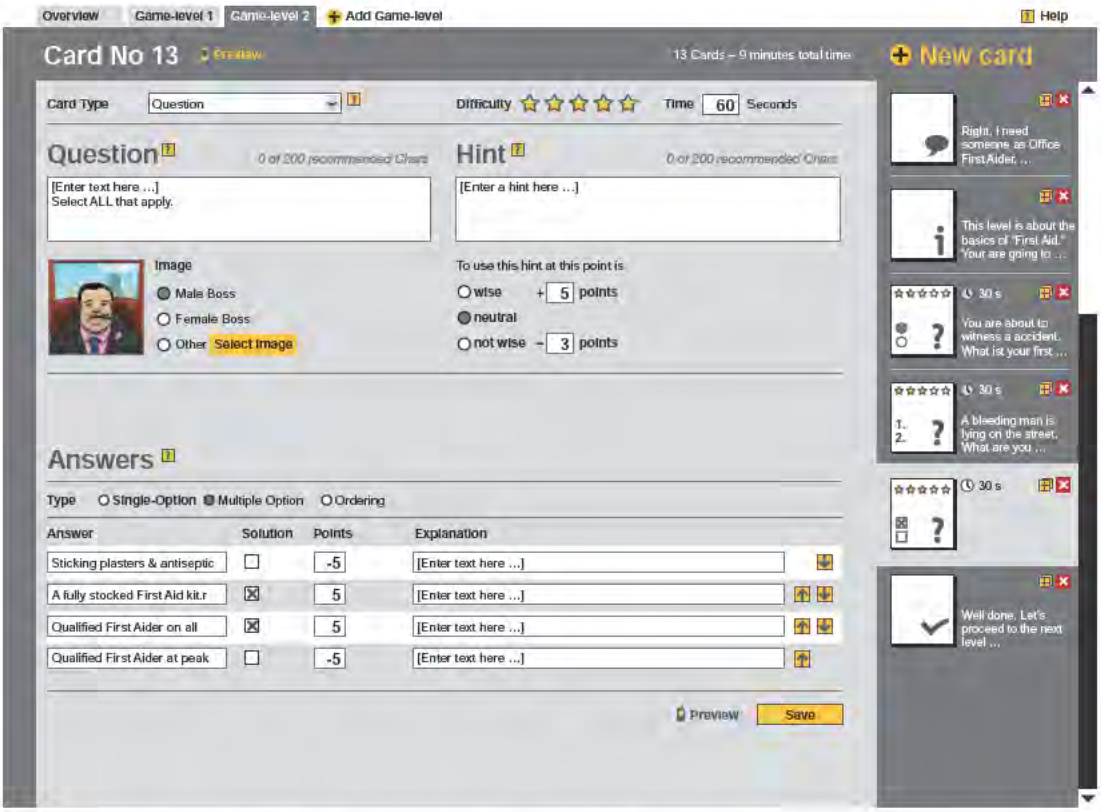


Figure C.2: Authoring card for a Multiple-Click Question.

FIFTH THIRD EX1062
PART 2 of 3
Page 267 of 379

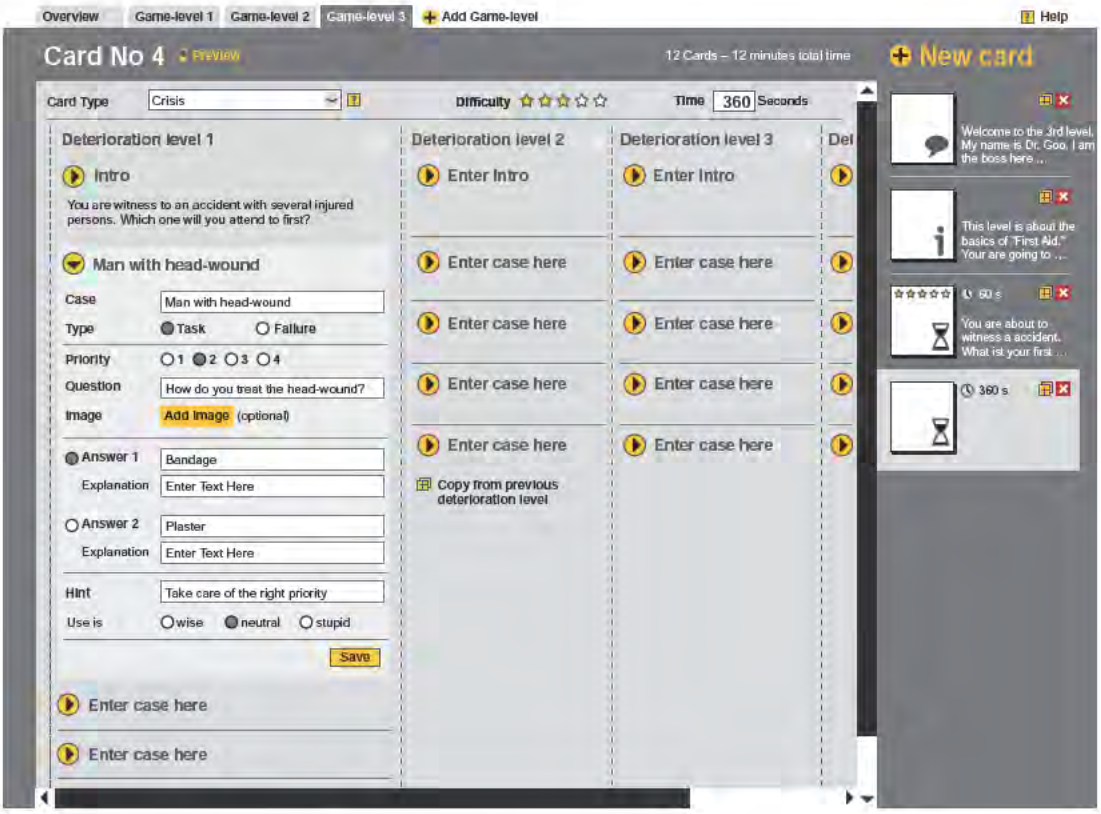


Figure C.3: Authoring interface for a simulation (partially collapsed fields to provide a panoramic view).

FIFTH THIRD EX1062
PART 2 of 3
Page 268 of 379

Appendix D

Test-Devices

The used test devices and their key characteristics are presented in figures D.1 to D.4.

FIFTH THIRD EX1062
PART 2 of 3
Page 269 of 379



Sony-Ericsson m600i

OS	Symbian OS v9.1, UIQ 3.0
Interactivity	TFT touchscreen
	QWERTY keyboard
	240 x 320 pixels
Connectivity	GPRS
	Bluetooth 2.0
	Infrared Port
Java	CLDC 1.1
	MIDP 2.0
	WMAPI 2.0
	MMAPI

Figure D.1: Specification of used test device



Sony-Ericsson v630i

OS	Proprietary
Interactivity	176 x 220 pixels
	256k colours
Connectivity	GPRS
	Bluetooth 2.0
Java	CLDC 1.1
	MIDP 2.0
	WMAPI 2.0
	MMAPI

Figure D.2: Specification of used test device

FIFTH THIRD EX1062
PART 2 of 3
Page 270 of 379



Nokia 6230i

OS	Proprietary
Interactivity	240 x 320 pixels 65k colours
Connectivity	GPRS Bluetooth 1.2 Infrared Port
Java	CLDC 1.1 MIDP 2.0 WMAPI MMAPI 1.1

Figure D.3: Specification of used test device



Sony-Ericsson P990i

OS	Symbian OS v9.1, UIQ 3.0
Interactivity	TFT touchscreen QWERTY keyboard 240 x 320 pixels
Connectivity	GPRS Bluetooth 2.0 Infrared Port
Java	CLDC 1.1 MIDP 2.0 WMAPI 2.0 MMAPI

Figure D.4: Specification of used test device

FIFTH THIRD EX1062
PART 2 of 3
Page 271 of 379

APPENDIX
HARRISON18

Advertisement



Embedded Computer Vision

Book | © 2009

Save book

Accessibility Information

Overview

Editors: Branislav Kisačanin, Shuvra S. Bhattacharyya, Sek Chai

- Provides historical perspective, the latest research results and a vision for future developments in this new field of embedded computer vision
- Contains high-level, state-of-the-art research results
- Looks ahead, providing a sense of what major applications could be expected in the near future

Part of the book series: Advances in Computer Vision and Pattern Recognition (ACVPR)

31k Accesses 28 Citations 9 Altmetric

This is a preview of subscription content, log in via an institution to check access.

About this book

As a graduate student at Ohio State in the mid-1970s, I inherited a unique c- puter vision laboratory from the doctoral research of previous students. They had designed and built an early frame-grabber to deliver digitized color video from a (very large) electronic video camera on a tripod to a mini-computer (sic) with a (huge!) disk drive—about the size of four washing machines. They had also - signed a binary image array processor and programming language, complete with a user's guide, to facilitate designing software for this one-of-a-kindprocessor. The overall system enabled programmable real-time image processing at video rate for many operations. I had the whole lab to myself. I designed software that detected an object in the

Access this book

Log in via an institution →

eBook USD 84.99
Price excludes VAT (USA)

- Available as PDF
- Read on any device
- Instant download
- Own it forever

Buy eBook →

Softcover Book USD 109.99

Hardcover Book USD 109.99

Tax calculation will be finalised at checkout

Other ways to access

Licence this eBook for your library →

Institutional subscriptions →

Sections

Overview

eldofview,trackeditsmovementsinrealtime,anddisplayedarunningdescription of the events in English. For example: "An object has appeared in the upper right corner...Itismovingdownandtotheleft...Nowtheobjectisgettingcloser...The object moved out of sight to the left"—about like that. The algorithms were simple, relying on a suf cient image intensity difference to separate the object from the background (a plain wall). From computer vision papers I had read, I knew that vision in general imaging conditions is much more sophisticated. But it worked, it was great fun, and I was hooked.

About this book

Table of contents (13 chapters)

Reviews

Similar content being viewed by others



Electronics Engineering Perspectives on Computer Vision Applications: An Overview of Techniques,...

Chapter © 2023



Embedded Vision Systems: A Review of the Literature

Chapter © 2018



Hardware architecture exploration: automatic exploration of distributed automotive hardware...

Article 12 March 2020

Explore related subjects

Discover the latest articles, books and news in related subjects.

Automated Pattern Recognition

Computer Vision

Computer Imaging, Vision, Pattern Recognition and Graphics

Internet of Things Applications in Smart Environments

Search within this book

Search

Table of contents (13 chapters)

Front Matter Download chapter PDF

Pages I-XXVIII

Introduction

Front Matter Download chapter PDF

Pages 1-1

Hardware Considerations for Embedded Vision Systems

Mathias Kölsch, Steven Butner

Pages 3-26

Design Methodology for Embedded Computer Vision Systems

Sankalita Saha, Shuvra S. Bhattacharyya
Pages 27-47

We Canwatch It For You Wholesale

Alan J. Lipton
Pages 49-76

Advances in Embedded Computer Vision

Front Matter

Download chapter PDF

Pages 77-77

Using Robust Local Features on DSP-Based Embedded Systems

Clemens Arth, Christian Leistner, Horst Bischof
Pages 79-100

Benchmarks of Low-Level Vision Algorithms for DSP, FPGA, and Mobile PC Processors

Daniel Baumgartner, Peter Roessler, Wilfried Kubinger, Christian Zinner, Karina Ambrosch
Pages 101-120

SAD-Based Stereo Matching Using FPGAs

Karina Ambrosch, Martin Humenberger, Wilfried Kubinger, Andreas Steininger
Pages 121-138

Motion History Histograms for Human Action Recognition

Hongying Meng, Nick Pears, Michael Freeman, Chris Bailey
Pages 139-162

Embedded Real-Time Surveillance Using Multimodal Mean Background Modeling

Senyo Apewokin, Brian Valentine, Dana Forsthoefel, Linda Wills, Scott Wills, Antonio Gentile
Pages 163-175

Implementation Considerations for Automotive Vision Systems on a Fixed-Point DSP

Zoran Nikolić
Pages 177-194

Towards OpenVL: Improving Real-Time Performance of Computer Vision Applications

Changsong Shen, James J. Little, Sidney Fels
Pages 195-216

Looking Ahead

Front Matter

Download chapter PDF

Pages 217-217

Mobile Challenges for Embedded Computer Vision

Sek Chai
Pages 219-235

Challenges in Video Analytics

Nikhil Gagvani
Pages 237-256

Challenges of Embedded Computer Vision in Automotive Safety Systems

Yan Zhang, Arnab S. Dhua, Stephen J. Kiselewich, William A. Bauson
Pages 257-279

Back Matter

Pages 281-282

Download chapter PDF

Back to top

Reviews

From the reviews:
"The book is a result of the Embedded Computer Vision Workshop 2007. ... provides a very good overview of the current state of the art in embedded computer vision and of the major trends and growing markets. ... it is a good start and provides an extensive list of references to look for if one wants to go into more detail. Overall I would recommend this book to anyone interested in getting into this exciting field." (Marcus E. Hennecke, IAPR Newsletter, Vol. 33 (2), April, 2011)

Editors and Affiliations

Texas Instruments, Dallas, USA
Branislav Kisačanin

University of Maryland, College Park, USA
Shuvra S. Bhattacharyya

Motorola, Schaumburg, USA
Sek Chai

Accessibility Information

Accessibility information for this book is coming soon. We're working to make it available as quickly as possible. Thank you for your patience.

Bibliographic Information

Book Title
Embedded Computer Vision

Editors
Branislav Kisačanin, Shuvra S. Bhattacharyya, Sek Chai

Series Title
Advances in Computer Vision and Pattern Recognition

DOI
https://doi.org/10.1007/978-

Publisher
Springer London

eBook Packages
Computer Science, Computer

1-84800-304-0

Science (R0)

Copyright Information
Springer-Verlag London 2009

Hardcover ISBN
978-1-84800-303-3
Published: 06 October 2008

Softcover ISBN
978-1-84996-776-1
Published: 21 October 2010

eBook ISBN
978-1-84800-304-0
Published: 26 September 2008

Series ISSN
2191-6586

Series E-ISSN
2191-6594

Edition Number
1

Number of Pages
XXVIII, 284

Topics
Image Processing and Computer Vision, Pattern Recognition, Computer Imaging, Vision, Pattern Recognition and Graphics

Keywords

Automotive Safety · Computer Vision · Driver Assistance · Embedded Computer Vision · Embedded Systems · Pattern Recognition · Video Analytics · Video Surveillance · action recognition · human action recognition

Publish with us

Policies and ethics

Back to top

Discover content
Journals A-Z
Books A-Z
Subjects A-Z

Publish with us
Journal finder
Publish your research
Language editing
Open access publishing

Products and services
Our products
Librarians
Societies
Partners and advertisers

Our brands
Springer
Nature Portfolio
BMC
Palgrave Macmillan
Apress
Discover

Your privacy choices/Manage cookies · Your US state privacy rights · Accessibility statement · Terms and conditions · Privacy policy · Help and support · Legal notice · Cancel contracts here

74.98.214.120
Not affiliated

SPRINGER NATURE
© 2026 Springer Nature

Advertisement

Mobile Challenges for Embedded Computer Vision

Chapter

pp 219–235 | Cite this chapter

Save chapter



Embedded Computer Vision

Sek Chai

Part of the book series: Advances in Computer Vision and Pattern Recognition ((ACVPR))

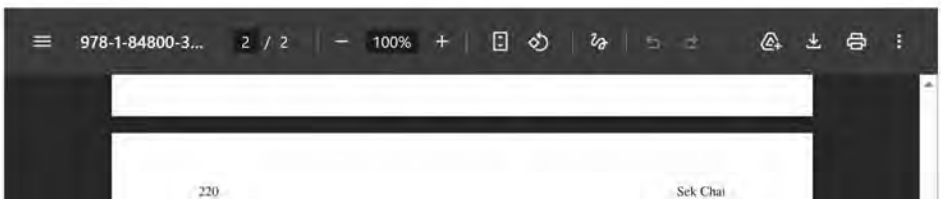
2374 Accesses 7 Citations

Abstract

The mobile environment poses uniquely challenging constraints for designers of embedded computer vision systems. There are traditional issues such as size, weight, and power, which are readily evident. However, there are also other less tangible obstacles related to technology acceptance and business models that stand in the way of a successful product deployment. In this chapter, I describe these issues as well as other qualities desired in a mobile smart camera using computer vision algorithms to "see and understand" the scene. The target platform of discussion is the mobile handset, as this platform is poised to be the ubiquitous consumer device all around the world.

This is a preview of subscription content, log in via an institution to check access.

Preview



Access this chapter

Log in via an institution

Subscribe and save

Springer+ from $39.99 /Month

Starting from 10 chapters or articles per month

Softcover Book USD 109.99

Hardcover Book USD 109.99

Tax calculation will be finalised at checkout

Purchases are for personal use only

Institutional subscriptions

Sections References

Abstract

Preview

References

Author information

Editor information

Rights and permissions

Copyright information

In embedded portable devices such as mobile camera phones, digital cameras, and camcorders, the elevated performance has to be delivered with limited size, weight, and power. These constraints directly correlate to the performance of the smart camera because size, weight, and power limit the type of optics, lighting, and processor that can be integrated. The main task for designers would therefore involve the selection of the best components for the smart camera while meeting these technical constraints.

In addition to technical challenges, embedded smart cameras face other intangible barriers in deployment. The barriers range from the very basic issues such as user acceptance of new technology to the more complicated issues such as having a good business model for the application. It is important to note that these factors play a major, if not principal, role in determining whether a consumer product using computer vision algorithms can be deployed successfully.

This chapter describes these challenges—both technical constraints and intangible barriers—in more detail. The goal is to motivate future designers by delineating issues that may arise. It is not intended as a cookbook or checklist because such an approach would more likely produce unsuccessful results for the readers. The overall focus of this chapter revolves around mobile handsets as the choice for an embedded system to integrate computer vision algorithms. While there have been numerous articles and books on the design of mobile handsets, the application that runs on it, and the wireless network it uses for communication, there is no comprehensive article about the use of computer vision on this platform.

The growth of the digital imaging industry is driven primarily by mobile handsets with embedded cameras, that is, camera-phones. First introduced in 2000, sales of camera phones have since surpassed sales of both film and digital cameras, and the entire digital imaging market continues to accelerate [1]. Today, imaging applications on camera phones involve the capturing, processing, and transmission of visual content on wireless mobile devices for images. In the next-generation camera-phones, these applications will demand higher resolution (multiple megapixels), better image quality through image finishing, and faster frame rates (video quality at 30 frames per second). Furthermore, designers will look towards new product features based on computer vision algorithms, much akin to those in smart cameras [2,3] in order to improve sales. The leading marketable feature, pixel resolution, has lost its momentum as consumers are already becoming satisfied with the quality of an image of several megapixels.

Section 11.2 provides an overview of applications. In Section 11.3, technology constraints for the mobile environment are discussed. Intangible obstacles are covered in Section 11.4. Finally, Section 11.5 concludes with future directions for mobile embedded computer vision systems.

About this chapter

Publish with us

Similar content being viewed by others



Image and video processing on mobile devices: a survey

Article 21 June 2021



CNN-Based Mobile Device Detection Using Still Images

Chapter © 2022



Realtime 3D Reconstruction with Mobile Devices

Chapter © 2019

Explore related subjects

Discover the latest articles, books and news in related subjects, suggested using machine learning.

Computer Vision | Embedded Systems | Mobile Computing | Mobile and Network Security | Object vision | Photography | Internet of Things Applications in Smart Environments

References

1. Brian O'Rourke, "CCDs & CMOS: Zooming in on the image sensor market," In-Stat Report IN030702MI, September 2003.

Google Scholar

2. D. S. Wills, J. M. Baker, Jr., H. H. Cat, S. M. Chai, L. Codrescu, J. Cruz-Rivera, J. C. Eble, A. Gentile, M. A. Hopper, W. S. Lacy, A. Löpez-Lagunas, P. May, S. Smith, and T. Taha, "Processing architecture for smart pixel systems," *IEEE J. Select Topics Quantum Electron*, v. 2, no 1, pp. 24-34, 1996.

Article Google Scholar

3. Wayne Wolf, Burak Ozer, Tiehan Lv, "Smart cameras as embedded systems," *IEEE Computer*, September 2002, pp. 48-53

Google Scholar

4. J. Adams, K. Parulski, and K. Spaulding, "Color processing in digital cameras," *IEEE Micro*, no. 18, pp. 20-30, 1998.

Article Google Scholar

5. Andrew Wilson, "Understanding camera performance specs," *Vision Systems Design*, vol 12, no 7, July 2007, pp. 39-45.

Google Scholar

6. Gregory K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, v. 34, no. 4, April 1991, pp.30-44.

Google Scholar

7. Didier Le Gall, "MPEG: a video compression standard for multimedia applications," *Communications of the ACM*, Special issue on digital multimedia systems, v. 34, no. 4, April 1991, pp. 46-58.

Google Scholar

8. Vasudev Bhaskaran, Konstantinos Konstantinides, *Image and Video Compression Standards*, 2nd edition, Kluwer Academic Press, 1997.

Google Scholar

9. Xi-Ping Luo, Jun Li, Li-Xin Zhen, "Design and implementation of a card reader based on built-in camera," *Proceedings of the 17th International Conference on Pattern Recognition*, v. 1, 23-26 Aug. 2004, pp. 417-420.

Google Scholar

10. J. Coughlan, R. Manduchi, "Color targets: fiducials to help visually impaired people find their way by camera phone," *EURASIP Journal on Image and Video Processing*, special issue on image and video processing for disability, v. 2007, article ID 96357, 2007.

Google Scholar

11. Scalado AB, Lund, Sweden, "Scalado heralds the dawn of a 'new age' for mobile imaging at the Mobile World Congress in Barcelona," press release, http://www.scalado.com/m4n.

12. Eyal de Lara, Maria Ebling, "New products: motion-sensing cell phones," *IEEE Pervasive Computing*, v 6, no 3, July-Sept. 2007, pp.15-17.

Google Scholar

13. M.Sohn, G. Lee, "ISeeU: Camera-based user interface for a handheld computer," *MobileHCI'05*, Sept 2005, pp. 299-302.

Google Scholar

14. Sony Computer Entertainment, Inc., Sony Eye Toy, www.eyetoy.com.

15. Kris Graft, "Analysis: history of cell-phone gaming," *Business Week*, January 22, 2006.

Google Scholar

16. Y. Cheng, M.W. Maimone, L. Matthies, "Visual odometry on the Mars exploration rovers – a tool to ensure accurate driving and science imaging," *IEEE Robotics & Automation Magazine*, v. 13, no. 2, June 2006, pp. 54-62.

Google Scholar

17. Roland T. Rust, Debora V. Thompson, RebeccaW. Hamilton, "Defeating feature fatigue," *Harvard Business Review*, Feb 1, 2006.

Google Scholar

18. D. Talla, J. Gobton, "Using DaVinci technology for digital video devices," *Computer*, v. 40, no.10, Oct. 2007, pp. 53-61.

Google Scholar

19. Max Baron, "Freescale's MXC voted best: the crown goes to Freescale's MXC91321 chip," *Microprocessor Report*, January 30, 2006, pp. 1-3.

Google Scholar

20. Tomas Akenine-Müller, Jacob Strüm, "Graphics for the masses: a hardware rasterization architecture for mobile phones," *ACM Transactions on Graphics (TOG)*, v. 22, no 3, July 2003, pp. 801-808.

Google Scholar

21. Pei Zheng, Lionel Ni, Lionel M. Ni, *Smart Phone and Next-Generation Mobile Computing*, Elsevier Science & Technology Books, December 2005.

Google Scholar

22. Alan Zeichick, "Look Ma, no wires," *NetNews*, v. 11, no. 4, December 2007, pp. 5-8.

Google Scholar

23. Richard Harrison, Mark Shackman, *Symbian OS C++ for Mobile Phones*, Symbian Press, Wiley, 2007.

Google Scholar

24. Tommi Mikkonen, *Programming Mobile Devices: An Introduction for Practitioners*, Wiley, 2007.

Google Scholar

25. J. Owens et al., "A survey of general-purpose computation on graphics hardware," *Proc. Eurographics*, 2005, pp. 21-51.

Google Scholar

26. S. M. Chai, et al., "Streaming processors for next-generation mobile imaging applications," *IEEE Communications Magazine*, Circuits for Communication Series, vol 43, no 12, Dec 2005, pp. 81-89.

Google Scholar

27. M. Cummings, S.Haruyama, "FPGA in the software radio," *IEEE Communications*, v. 37, no. 2, Feb 1999, pp. 108-112.

Google Scholar

28. T. Tuan, S. Kao, A. Rahman, S. Das, S. Trimberger, "A 90-nm low-power FPGA for battery-powered applications," *Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field-Programmable Gate Arrays*, Monterey, California, 2006, pp. 3-11.

Google Scholar

29. A. Löpez-Lagunas, S. M. Chai, "Memory bandwidth optimization through stream descriptors," *ACM SIGARCH Computer Architecture Newsletter*, vol 34, no 1, pp. 57-64, March 2006.

Google Scholar

30. S. Palacharla, R.E. Kessler, "Evaluating stream buffers as a secondary cache replacement," *Proceedings of the 21st Annual International Symposium on Computer Architecture*, pp. 24-33, April 1994.

Google Scholar

31. S. A. McKee, et. al., "Dynamic access ordering for streamed computations," *IEEE Transactions on Computers*, vol. 49, no. 11, november 2000.

Google Scholar

32. L. Zhang, Z. Fang, M. Parker, B. K. Mathew, L. Schaelicke, J. B. Carter, W. C. Hsieh, S. A. McKee, "The impulse memory controller," *IEEE Transactions on Computers*, pp. 1117-1132, nov 2001.

Google Scholar

33. A. Bellaouar, M. I. Elmasry, *Low-Power Digital VLSI Design: Circuits and Systems*, Springer, June 30, 1995.

Google Scholar

34. W. Bidermann, A. El Gamal, S. Ewedemi, J. Reyneri, H. Tian, D. Wile, D. Yang, "A 0.18 /spl mu/m high dynamic range NTSC/PAL imaging system-on-chip with embedded DRAM frame buffer," *IEEE International Solid-State Circuits Conference*, v.1, 2003, pp. 212-488.

Google Scholar

35. S. B. Gokturk, H. Yalcin, C. Bamji, "A time-of-flight depth sensor - system description, issues and solutions," *Computer Vision and Pattern Recognition Workshop*, June 2004, p. 35.

Google Scholar

36. Eugene Hecht. *Optics* (4th ed.). Pearson Education. 2001.

Google Scholar

37. N. Paragios, Y. Chen, and O. Faugeras, eds., *The Handbook of Mathematical Models in Computer Vision*, Springer, 2005.

Google Scholar

38. B. Berge, "Liquid lens technology: principle of electrowetting based lenses and applications to imaging," *Proc. IEEE International Conference on Micro Electro Mechanical Systems*, 2005.

Google Scholar

39. E. J. Tremblay, R. A. Stack, R. L. Morrison, and J. E. Ford, "Ultrathin cameras using annular folded optics," *Applied Optics*, vol. 46, Issue 4, pp. 463-471.

Google Scholar

40. Martin Buehler, Karl Iagnemma, and Sanjiv Singh, *The 2005 DARPA Grand Challenge: The Great Robot Race*, Springer, 2007.

Google Scholar

41. C. Lankshear, I. Snyder, *Teachers and Technoliteracy: Managing Literacy, Technology and Learning in Schools*, St. Leonards, NSW, Australia: Allen & Unwin, 2000.

Google Scholar

42. P. J. Phillips, M. Hyeonjoon, S.A. Rizvi, and P.J. Rauss, "The FERET evaluation methodology for face-recognition algorithms" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, no. 10, Oct. 2000, pp. 1090-1104.

Google Scholar

43. P. Courtney, N. A. Thacker, "Performance Characterization in Computer Vision." In *Imaging and Vision Systems*, Jacques Blanc-Talon and Dan Popescu (Eds.), noVA Science Books, 2001.

Google Scholar

44. Chunho Lee, Miodrag Potkonjak, William H. Mangione-Smith, "MediaBench: a tool for evaluating and synthesizing multimedia and communicatons systems," *Proceedings of the 30th annual ACM/IEEE International Symposium on Microarchitecture*, 1997, pp. 330-335.

Google Scholar

45. R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, A. Choudhary, "MineBench: A benchmark suite for data mining workloads," *2006 IEEE International Symposium on Workload Characterization*, Oct. 2006, pp. 182-188.

Google Scholar

46. OpenCV. http://www.intel.com/research/mrl/research/opencv/

47. Gary Bradski, Adrian Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, Inc., 2008.

Google Scholar

48. Petri Honkamaa, Jani Jäppinen, Charles Woodward, "A lightweight approach for augmented reality on camera phones using 2D images to simulate 3D," *Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia*, vol. 284, Oulu, Finland, 2007, pp. 155-159.

Google Scholar

49. SMIA: Standard Mobile Imaging Architecture, http://www.smia-forum.org.

50. Lee Nelson, "Solving the Problems of Mobile Imaging," *Advanced Imaging*, vol 22, no 4, April 2007, pp. 10-13.

Google Scholar

51. Clayton M. Christensen, *The Innovator's Dilemma: The Revolutionary Book that Will Change the Way You Do Business*, Collins, 2003.

Google Scholar

52. David Metcalf, *M-Learning: Mobile E-Learning*, HRD Press, Inc., January 2006.

Google Scholar

Download references

Author information

Authors and Affiliations

Motorola Labs, Schaumburg, IL, USA
Sek Chai

Editor information

Editors and Affiliations

Texas Instruments, Dallas, TX, USA
Branislav Kisačanin PhD

University of Maryland, College Park, MD, USA
Shuvra S. Bhattacharyya PhD

Motorola, Schaumburg, IL, USA
Sek Chai PhD

Rights and permissions

Reprints and permissions

Copyright information

© 2009 Springer-Verlag London Limited

About this chapter

Cite this chapter

Chai, S. (2009). Mobile Challenges for Embedded Computer Vision. In: Kisačanin, B., Bhattacharyya, S.S., Chai, S. (eds) Embedded Computer Vision. Advances in Computer Vision and Pattern Recognition. Springer, London. https://doi.org/10.1007/978-1-84800-304-0_11

Download citation

.RIS .ENW .BIB

DOI	Publisher Name	Print ISBN
https://doi.org/10.1007/978-1-84800-304-0_11	Springer, London	978-1-84800-303-3

Online ISBN	eBook Packages

978-1-84800-304-0

Computer Science
Computer Science (R0)

Keywords

Computer Vision | Graphical Processing Unit | Image Sensor | Camera Phone | Computer Vision Application

These keywords were added by machine and not by the authors. This process is experimental and the keywords may be updated as the learning algorithm improves.

Publish with us

Policies and ethics

Discover content
Journals A-Z
Books A-Z
Subjects A-Z

Publish with us
Journal finder
Publish your research
Language editing
Open access publishing

Products and services
Our products
Librarians
Societies
Partners and advertisers

Our brands
Springer
Nature Portfolio
BMC
Palgrave Macmillan
Apress
Discover

Your privacy choices/Manage cookies | Your US state privacy rights | Accessibility statement | Terms and conditions | Privacy policy | Help and support
Legal notice | Cancel contracts here

74.98.214.120
Not affiliated

SPRINGER NATURE
© 2026 Springer Nature

https://web.archive.org/web/20100000000000*/http://www.uweb.ucsb.edu/~yichuwang/ecv/paper/mobile_challenges_for_ecv.pdf July 30, 2026

INTERNET ARCHIVE WEB TEXTS VIDEO AUDIO SOFTWARE All Search SIGN UP | LOG IN UPLOAD

ABOUT BLOG EVENTS PROJECTS HELP DONATE CONTACT JOBS VOLUNTEER

DONATE

INTERNET ARCHIVE WaybackMachine

Explore more than 1 trillion web pages saved over time

http://www.uweb.ucsb.edu/~yichuwang/ecv/paper/mobile_challer

Calendar · Collections · Changes · Summary · Site Map · URLs

Saved **1 time** August 14, 2010

003 2004 2005 2006 2007 2008 2009 **2010** 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026

JAN

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

FEB

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28						

MAR

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

APR

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

MAY

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

JUN

		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

JUL

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

AUG

1	2	3	4	5	6	7
8	9	10	11	12	13	**14**
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

SEP

		1	2	3	4	
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

OCT

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

NOV

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

DEC

			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Note

This calendar view maps the number of times **http://www.uweb.ucsb.edu/~yichuwang/ecv/paper/mobile_challenges_for_ecv.pdf** was crawled by the Wayback Machine, *not* how many times the site was actually updated. More info in the FAQ.

FAQ | Contact Us | Terms of Service (Dec 31, 2014)

The Wayback Machine is an initiative of the Internet Archive, a 501(c)(3) non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. Other projects include Open Library & archive-it.org.

Your use of the Wayback Machine is subject to the Internet Archive's Terms of Use

https://web.archive.org/web/20100814145849/http://www.uweb.ucsb.edu/~yichuwang/ecv/paper/mobile_challenges_for_ecv.pdf July 30, 2026
INTERNET ARCHIVE Wayback Machine http://www.uweb.ucsb.edu/~yichuwang/ecv/paper/mobile_challenges_for_ecv.pdf Go JUL AUG SEP 14 2010 2009 2011 About this capture
1 capture 14 Aug 2010
mobile_challenges_for_ecv.pdf 1 / 17 100%

Chapter 11
Mobile Challenges for Embedded Computer Vision

Sek Chai

Abstract The mobile environment poses uniquely challenging constraints for designers of embedded computer vision systems. There are traditional issues such as size, weight, and power, which are readily evident. However, there are also other less tangible obstacles related to technology acceptance and business models that stand in the way of a successful product deployment. In this chapter, I describe these issues as well as other qualities desired in a mobile smart camera using computer vision algorithms to "see and understand" the scene. The target platform of discussion is the mobile handset, as this platform is poised to be the ubiquitous consumer device all around the world.

Chapter 11
Mobile Challenges for Embedded Computer Vision

Sek Chai

Abstract The mobile environment poses uniquely challenging constraints for designers of embedded computer vision systems. There are traditional issues such as size, weight, and power, which are readily evident. However, there are also other less tangible obstacles related to technology acceptance and business models that stand in the way of a successful product deployment. In this chapter, I describe these issues as well as other qualities desired in a mobile smart camera using computer vision algorithms to "see and understand" the scene. The target platform of discussion is the mobile handset, as this platform is poised to be the ubiquitous consumer device all around the world.

11.1 Introduction

Computer vision is a branch of computer science concerned with analyzing images to extract information about the environment. Much like the human visual system, embedded computer vision systems perform these same visual functions in a wide variety of products. They enable automation, speed, and precision for tasks that were previously manual.

Embedded smart cameras using computer vision algorithms are complex systems integrating an assortment of components such as lens, image sensors, processors, and memories. Designing such a system requires a team of experts with multidisciplinary knowledge ranging from optics to computer architecture in order to deliver products that run the algorithms. Unlike today's imaging applications that involve well-understood algorithms for image finishing and compression, computer vision algorithms embody elements of perception and intelligence that can require processing power in the range of supercomputers.

Sek Chai
Motorola Labs, Schaumburg, IL, USA, e-mail: sek.chai@motorola.com

FIFTH THIRD EX1062
PART 2 of 3
Page 290 of 379

In embedded portable devices such as mobile camera phones, digital cameras, and camcorders, the elevated performance has to be delivered with limited size, weight, and power. These constraints directly correlate to the performance of the smart camera because size, weight, and power limit the type of optics, lighting, and processor that can be integrated. The main task for designers would therefore involve the selection of the best components for the smart camera while meeting these technical constraints.

In addition to technical challenges, embedded smart cameras face other intangible barriers in deployment. The barriers range from the very basic issues such as user acceptance of new technology to the more complicated issues such as having a good business model for the application. It is important to note that these factors play a major, if not principal, role in determining whether a consumer product using computer vision algorithms can be deployed successfully.

This chapter describes these challenges—both technical constraints and intangible barriers—in more detail. The goal is to motivate future designers by delineating issues that may arise. It is not intended as a cookbook or checklist because such an approach would more likely produce unsuccessful results for the readers. The overall focus of this chapter revolves around mobile handsets as the choice for an embedded system to integrate computer vision algorithms. While there have been numerous articles and books on the design of mobile handsets, the application that runs on it, and the wireless network it uses for communication, there is no comprehensive article about the use of computer vision on this platform.

The growth of the digital imaging industry is driven primarily by mobile handsets with embedded cameras, that is, camera-phones. First introduced in 2000, sales of camera phones have since surpassed sales of both film and digital cameras, and the entire digital imaging market continues to accelerate [1]. Today, imaging applications on camera phones involve the capturing, processing, and transmission of visual content on wireless mobile devices for images. In the next-generation camera-phones, these applications will demand higher resolution (multiple megapixels), better image quality through image finishing, and faster frame rates (video quality at 30 frames per second). Furthermore, designers will look towards new product features based on computer vision algorithms, much akin to those in smart cameras [2,3] in order to improve sales. The leading marketable feature, pixel resolution, has lost its momentum as consumers are already becoming satisfied with the quality of an image of several megapixels.

Section 11.2 provides an overview of applications. In Section 11.3, technology constraints for the mobile environment are discussed. Intangible obstacles are covered in Section 11.4. Finally, Section 11.5 concludes with future directions for mobile embedded computer vision systems.

FIFTH THIRD EX1062
PART 2 of 3
Page 291 of 379

11.2 In Search of the Killer Applications

This section briefly describes imaging applications in mobile handsets in order to highlight the computation and memory access characteristics. The aim is to show both computational and memory requirements for these applications. It motivates the designer to consider parallel processing elements and efficient memory access in order to manage applications for embedded computer vision.

Today's mobile imaging applications include color interpolation, white balance, gamma correction, and compression/decompression. For the sub megapixel resolutions such as VGA-sized images (640×480), these functions can be easily handled by off-the-shelf imaging DSPs (digital signal processors) with performance ranging up to 500 MOPs (million operations per seconds). Next-generation mobile imaging applications will be the same but for higher-resolution images and video frame rates. Furthermore, computer vision algorithms will be used to enable smart camera features. These next-generation applications will demand more resolution and smart camera features that can exceed more than twice the performance of an imaging DSP [2,3].

11.2.1 Image Finishing

Image finishing consists of a series of processing steps to render raw sensor data into images pleasing to the human eye [4]. They consist of a variety of algorithms that include: color interpolation, white balance, gamma correction, and compression. Virtually all mobile imaging devices implement these algorithms either in image sensor companion chips or in application processors.

Color interpolation is used to convert a single sensor's output from a mosaic of colored pixels into a full-color image. The mosaic is typically organized in such a way that images are captured in an interlaced RGB (red, green, blue) pattern. The process involves finding the correct color value for the missing pixel located among several other pixels. The algorithm includes filtering operations over a range of 3×3 to 5×5 tiles for the three RGB color planes.

White balance and gamma correction are steps to tune the image pixel data to match the intended color range. For example, a white color should be visually white, with RGB colors saturated for their range of values. However, the image sensor does not necessarily produce ideal color ranges due to ambient noise, temperature, and dark currents [5]. Processing is needed to redistribute and scale the energy content of the RGB values. The algorithms typically include 3×3 matrix multiplications of pixel data with a filter and look-up-tables (LUT) to map the result to a new nonlinear color space.

Finally, compression standards such as JPEG [6], include steps to convert the RGB color space into a YCbCr color space with luminance (Y), red chrominance (Cr), and blue chrominance (Cb), before the compression procedure. The compression algorithm involves block comparisons, typically 8×8, and LUT to encode the

FIFTH THIRD EX1062
PART 2 of 3
Page 292 of 379

image data. Memory access patterns are 2D blocks based on the filter tile sizes (3×3 or 5×5 for color interpolation and 8×8 for JPEG compression).

11.2.2 Video Codec

MPEG compression standards [7] are widely used in digital video encoding products such as digital camcorders. Video messaging and two-way video conferencing are already available in camera phones today, but popular demand for such services takes time to foster. Wider-bandwidth networks, driven by content providers' eagerness to create a new market in mobile devices, will enable new applications such as video-on-demand and mobile TV broadcast. Content from home entertainment systems can be seamlessly shown on mobile devices, enabling user access in addition to mobility.

In general, the encoding process begins with preprocessing, which may include format conversion to convert the image into YCbCr color space and resizing to a desired image size. Motion estimation, 2D DCT, quantization, and entropy coding are the next stages in the coding process. For a CIF-sized video (352×240), the encoding process requires more than 1000 MOPs, which is beyond the capabilities of today's single embedded scalar processors [8]. The decoding processing follows a simpler inverse procedure and requires only 200 MOPs (easily achievable in scalar processors). Memory access patterns are based on macro-block and DCT filter sizes, which are typically 8×8 over the entire image.

11.2.3 Computer Vision

Smart cameras use computer vision algorithms to "see and understand" the scene. The "see" portion uses computer vision algorithms to segment, classify, and identify objects; the "understand" portion includes more complex learning algorithms to model and track objects. There are already application examples in security, entertainment, safety, and transportation, but not all of these applications apply to camera-phones. In general, these applications have the following abilities: to extract features and structures from images, and to derive semantic meaning in the form of metadata through a series of pattern-matching operations.

Computer vision algorithms identify objects in a scene, and consequently produce one region of an image that has more importance than other regions of the image. For example, in body gesture recognition [3], the region of the video image that includes portions of the hand (a region of interest, or ROI) may be studied to find the hand posture over a series of frames. A video image may also be processed to track different parts of the body. Information about detected objects is stored in data structure with fields containing features and location. Each object lies in a different region of the captured images which must be processed simultaneously to

FIFTH THIRD EX1062
PART 2 of 3
Page 293 of 379

find the overall effect of the body gesture. Memory access patterns are dependent on object size in the image, ranging from a few hundred to thousands of pixels in order to have enough resolution for feature detection. Furthermore, the object moves in subsequent frames, which corresponds to fetching a "moving" block of pixels from memory. That is, the memory access patterns would shift in memory addresses as the tracked object moves in a series of captured image frames.

11.2.4 Example Applications

While there are mobile imaging applications using aspects of computer vision, none has risen to a "killer app" status. Early example applications include business card readers [9] and aids for the visually impaired [10]. To be considered a "killer app," the application must be so compelling that consumers will find it absolutely necessary to have the feature. Such an application would surpass all of its competitive alternatives in consumer demand and popularity.

An application area that is growing in importance is image editing and manipulation. As users take more pictures using their camera phone, they also want to edit the captured images to improve its presentation. For example, users may want to change the contrast and brightness, or simply crop away portions of the image. Rather than performing these edits on a computer, whereby pictures must be downloaded beforehand, users can do so directly on their camera phone. This capability would enable user experiences that improve the spontaneity of photo-taking and sharing them at the same instant. Without this feature, many users may not even bother because the download process is too cumbersome.

In many cases, the editing process can be enhanced and automated by using computer vision algorithms to understand objects in the image. For example, object and face detection can simplify photo album creation by properly labeling photos as they are taken. Cropping and framing can be done to properly center the image on a face. Furthermore, panoramic images can be created by stitching together multiple images taken from different viewpoints. There are now companies that are marketing and licensing niche intellectual property and software with these features [11].

In addition to image editing, another application area is user interaction enhancements, whereby mobile handset users can attain a better experience in using the device because of an embedded camera running computer vision algorithms. These applications deal with supplementing or completely augmenting the way user interacts with the device. In addition to picture-taking, the camera's role is extended to input devices such as keyboards, joysticks, or touch screens. Computer vision algorithms enable the camera to function as a visual sensor to detect body and hand gestures [3,12].

A good example that illustrates this capability is map reading, where users would need to scroll quickly to different map regions or zoom into different areas [13]. Instead of repeatedly pressing keypads to perform the scroll and zoom, a smart camera application would track the user's face in relation to the onboard camera.

FIFTH THIRD EX1062
PART 2 of 3
Page 294 of 379

The effect is a fluid and intuitive movement of the displayed map region, resulting in a shorter time to complete the task of map reading.

Videogames are also examples where repetitive key presses can be replaced with object tracking using computer vision. Camera-based toys, such as the Sony EyeToy™, are a good example of the use of motion tracking and color recognition to effect game control [14]. Although promising, today's video games on mobile handsets lags behind such handhelds as the Sony PSP™and Nintendo DS™, due to the stark difference in processing power. Furthermore, the difficulty in advertising and delivering mobile games are some of the biggest reasons behind the growth in mobile gaming [15].

There are many other application areas that would benefit from the use of computer vision. They range from security (e.g., face detection), surveillance (e.g., person tracking), marketing/advertising (visual search) to health monitoring. While there are no killer apps to date, future designers should be encouraged, as the field looks promising and primed for future success.

11.3 Technology Constraints

This section describes some of the technical issues in mobile computer vision systems. The aim is to show both the traditional issues with embedded system design, and also issues specific to integrating computer vision applications in mobile handsets.

11.3.1 The Mobile Handset

Few consumer products match mobile handsets in the level of application integration. Mobile handsets not only make phone calls, but also take pictures and play music. They serve as a personal digital assistant (PDA), a game platform, a text-messaging system, and a television unit. Each generation of mobile handset brings new features as designers jockey for market position. Designers must meet the pressure of lowering the bill of materials (BOM) in order to deliver low product average selling price (ASP). This challenge comes amongst the increasingly growing set of features in a single product. This "feature creep" aspect is driven by both consumer demand and marketing.

Ever since the inception of the mobile handset in 1973, there has been continuous momentum to combine useful features for the consumer. Fig. 11.1 shows the timeline of key mobile phone introduction that intertwines with those of the digital camera timelines to bring us camera phones. The smart camera timeline has evolved from robotics, such as the Mars Exploration Rovers [16], to consumer products such as the Sony EyeToy™[14]. In the not too distant future, the mobile handset

FIFTH THIRD EX1062
PART 2 of 3
Page 295 of 379

timeline might intersect with the smart camera timeline, producing an integrated product using computer vision algorithms.

Additional features must be balanced with device usability. It is important to note that people may not always value an overly high-featured product. In a series of studies measuring what people value in selecting a product and later after using the product, users have been found to prefer a much simpler device [17]. The research has concluded that "even the tech-savvy suffered from this 'feature fatigue' effect." When considering smart camera functionality in mobile handsets, computer vision applications should be designed to improve device usability and not add to "feature fatigue."



Fig. 11.1 Mobile handset timeline intersects digital camera and computer vision timelines.

In addition to application integration, the mobile handset is a marvel of component integration. The basic parts are comprised of the following: a small microphone, a speaker, an LCD display, a keypad, antenna, battery. There are also chips to convert signals from analog to digital and vice versa. The heart of the phone is the application processor that handles the housekeeping and user interface. A digital signal processor (DSP) handles the signal processing for RF (radio frequency) amplifiers that transmit and receive signals through the antenna.

Great efforts have been made to integrate these components to single-chip solutions in order to reduce area on the circuit board, power dissipation, and overall system cost. Semiconductor vendors now offer application processor with DSPs as well as on-chip memory [18,19]. A graphical processing unit (GPU) [20] is used to display application content to the user. The reader is referred to [21] for more details about the mobile handset. The camera module for the mobile handset comes in an

FIFTH THIRD EX1062
PART 2 of 3
Page 296 of 379

integrated module with lens, image sensor, and companion chips. The companion chips are signal processing chips programmed with software to perform image finishing and compression. The finished image is transferred directly to the application processor for storage and further processing.

11.3.2 Computing Platform

Computer vision algorithms can reside in many different computing platforms on the mobile handset: application processor, DSP, GPU, camera-module companion chips. It is not unusual for existing applications to consume all of the processing horsepower in all of these computing platforms at once. For example, in an application such as video teleconferencing, the application processor may be running video codecs (coding and decoding of video streams), the DSP may be handling video transmission and reception, the GPU is rendering the image, and the camera-module companion chip is finishing the captured image.

In order to consider computer vision algorithms, the designer should consider how it would be integrated into the overall user experience. For example, in an operating scenario where a normal voice call is being made, the application processor may be lightly loaded, making it suitable to run other applications. In another example where the user is browsing the web, the camera-module companion chip may be lightly loaded or not used at all. It is important to make the computer vision application run seamlessly alongside existing applications. Otherwise, user acceptance would be low when the overall user experience suffers.

Today, the application processor would be the most acceptable computing platform to run the computer vision application as mobile handset makers are opening this platform to third-party software developers [22,23]. The GPU, DSP, or camera-module companion chip usually contains firmware that is not accessible to the public. Most third-party software is written in Java [24], with some support for native level drivers. Software libraries and application programmer interface (API) are made available in software developers kit (SDK). In most cases, software developers are given access to a frame buffer containing the image captured by the camera module. There is little to no support for designers to specify parameters in the camera module such as frame rate and exposure time [5]. This means that current computer vision applications are usually relegated to use image sensor settings that are optimized for picture taking. In the future, when the computing and camera module are part of an open platform, designers will have more flexibility to program and control the components.

It is also not uncharacteristic to consider extra processing hardware specifically for running computer vision algorithms. The hardware can come as a separate chipset or in the form of hardware accelerators in the application processor. There is already precedence: in the gaming application, graphics are accelerated on the GPU (a chip specifically made for that purpose) [25]; in video playback applications, DFT hardware engines [18,19] accelerate portions of video codecs. In addition,

FIFTH THIRD EX1062
PART 2 of 3
Page 297 of 379

specialized streaming processors [26] and low-power FPGA/CPLD [27,28] have been proposed for computer vision, but integration of an extra computing platform for computer vision would add cost and increase ASP. It can only be justified when there is a compelling application, or "killer app," that consumer demands.

11.3.3 Memory

In addition to computational horsepower needed by the computer vision algorithms, the designer should also consider memory bandwidth and memory allocation during early stages of the design process. These items are often considered as a design afterthought, which may cause the application to run slower than expected. This could result in poor device usability.

Section 11.2 of this chapter already provides some example memory access patterns for different applications such as image finishing, codecs, and computer vision. While still image processing consumes a small amount of bandwidth and allocated memory, video can be considerably demanding on today's memory subsystem. At the other end of the spectrum, memory subsystem design for computer vision algorithms can be extremely challenging because of the extra number of processing steps required to detect and classify objects.

There is promising research in memory controller designs to improve the memory bandwidth issues for streaming data [29]. Because computer vision data have predictable access patterns such as streaming data, the transfer of data from external memory to processor can be made more efficient. In [30], stream buffers are used to separate stream data from cached data because stream data have low temporal locality and can trash cache contents. In [31], a stream memory controller prefetches and buffers stream data from memory by dynamically reordering the accesses. In [32], data from sparse DRAM locations are prefetched and assembled as a contiguous cache line using shadow addresses (memory ranges not backed by DRAM). With a stream memory controller, system performance becomes dependent on average memory subsystem's bandwidth with less sensitivity to peak latency to access a data element. The total bus traffic is reduced and large caches are not necessary.

11.3.4 Power Consumption

Power dissipation is an important consumer metric for mobile handsets as it dictates the usage time (talk time, Internet use time, video playback time, audio playback time, etc.) and standby time. It is obvious that the longer the usage and standby time, the more desirable the device. At the same time, there is an opportunity to reduce the size and weight of the battery to achieve the same usage and standby time.

FIFTH THIRD EX1062
PART 2 of 3
Page 298 of 379

Mobile handsets have low-power consumption while operating (much less than desktop and laptops), and an almost negligible standby power when the device is not in use. This is evident in the drive for low power designs in the application processors [18,19,33]. Consequently, designers should pay attention to the energy budget in the battery and not expect a computer vision algorithm to run continuously. To save power, for example, designers may consider turning off the camera module when it is not needed or lowering the frame rate when the desired performance is not needed.

11.3.5 Cost and Performance

Mobile handsets can be separated into three categories[1]: low-cost, mid-tier, and smart phones. The lower-end phones are low-cost, high-volume devices with little to no features, except the ability to make phone calls. These low-cost phones may not even include a camera. The mid-tier phones are mid-range in prices with standard features such as a megapixel camera. They may be targeted toward the teens and tweens, and may have a music player for music enthusiasts.

Smart phones offer advanced capabilities much like a portable computer. Example smart phones include Nokia N95™, Apple IPhone™, RIM Blackberry™, and Motorola Q™. These phones are decked out with features such as PDA functions, large displays, and high-resolution (multi-megapixel) camera. They are targeted to the tech-savvy and also the business professionals, that is, those who can afford the premium cost. New computer vision applications are likely to first appear in smart phones. These mobile handsets have higher performance computing platform that can handle the extra application load. Furthermore, the better cameras provide better resolution and higher quality images for computer vision algorithms.

11.3.6 Image Sensor

The image sensors in today's mobile handset are primarily CMOS (complementary metal oxide semiconductor) image sensors. In comparison to CCD (charged-coupled devices), they have risen in quality and lowered in price in recent years [1], making them a good fit for the price-conscious mobile handset market. The resolution of an image sensor determines the amount of information that can be captured from a single shot. The higher the resolution, the more details at edges and the sharper the image in digital zooming.

Pixel resolution has been the leading marketable feature for camera phone sales. It is losing its momentum as most consumers are satisfied with the quality of an

[1] The categorizations and descriptions are generalizations to simplify discussions in this chapter. DynaTAC™, StarTAC™, Motorola Razr™, MotoMing™, and Motorola Q™are trademarks of Motorola Inc. Other product names are the property of their respective owners.

FIFTH THIRD EX1062
PART 2 of 3
Page 299 of 379

image at several megapixels. At those resolutions, the picture looks crisp, especially for spontaneous and casual snapshots. For archival or printing purposes, higher resolution may be desired.

As a result of market demand, semiconductor manufacturers have been optimizing image sensor designs for the mobile handset market. The large number of pixels in a single chip has been enabled by reducing individual pixel size. This is done to improve yield and cost of the imager. However, the sensitivity of each pixel is reduced because there is less area in each pixel to collect light. The dynamic range is also lessened because the pixel is collecting a smaller amount of light over time. That is, the saturation capacity of each pixel is reduced. To compensate, cameras increase the shutter speed to allow a smaller amount of light at one time, but produce images with less depth of field. This is equivalent to reducing the camera aperture. Finally, smaller pixels may be more subjective to dark noise, which is measurement of the variation in the number of electrons that are thermally generated within the imager [5]. Computer vision algorithms must then counteract the image sensor that is optimized for picture taking in a mobile handset. This may mean additional filters and processing stages to compensate for image sensors with smaller pixels.

There is ongoing research to develop better, more responsive image sensors [34]. These new approaches may enable or facilitate how computer vision algorithms are designed. For example, the image sensor may provide means to extract depth information [35], thereby reducing processing steps in the algorithm. Designers should also pay attention to image sensor research that is applied to other industries. Image sensors for the automotive and surveillance industry are typically more light-sensitive, have higher dynamic range, and are able to tolerate varying temperature ranges.

11.3.7 Illumination and Optics

Illumination and optics play a significant role in the camera operation and computer vision algorithm performance. Without sufficient illumination, the object of interest cannot be detected by the camera; without good optics, the object of interest may be blurred and difficult to detect. Extra processing steps may be required for image segmentation. This section will offer a cursory treatment of the subject area as a comprehensive discussion is beyond the scope of this chapter or book. The reader is directed to other texts for details on the theory and definitions of the terms related to measurement of light [36,37].

Designers should ask if additional lighting sources are needed in order for the application to work properly. The mobile handset offers only limited choices of active illumination. There may be an LED (light emitting diode) lighting source that is pulsed to provide constant light intensity while image is being captured. In higher-end camera phones, there may be a xenon strobe to illuminate the subject. There is also a battery drain issue to keep the illumination active. Designers should

FIFTH THIRD EX1062
PART 2 of 3
Page 300 of 379

understand these issues when developing their algorithm because lighting sources in a laboratory setting are well placed and the subject is adequately lit.

Most camera modules use lenses with short, fixed focal lengths. The focal length is the distance between the optical center of the lens to the image sensor where the image is focused. A short focal length means that the camera is set up for wide area shots rather than a deep one. In most cases, it is best to keep the subject within three to four feet of the camera. While there are options for movable optical lens for autofocus and other features, the added cost and weight make these options unsuitable for the mobile handset market.

There is active research aimed at producing new lens products for camera phones. For example, Varioptic's liquid-based lens offer variable focal lengths by changing the curvature of the liquid material in their lens without any moving parts [38]. Other approaches include hybrid lenses that bend and refocus light through a series of small mirrors within the lens [39].

11.4 Intangible Obstacles

This section describes some of the less technical issues in mobile computer vision systems. These issues are "intangible" because they deal with human psychology, measurability, and business models. The aim of this section is to show how such issues can influence the productization of computer vision applications in mobile handsets. It brings to light that even the best technical solution may not produce a successful product if these intangible obstacles are not addressed.

11.4.1 User Perception and Attitudes Towards Computer Vision

Computer vision is new technology that is being applied in many consumer products. Even though there has been very recent news about the success of computer vision [16,40], most mobile handset users do not understand the technical details behind the application. They are more likely to perceive difficulties in performing a task that is augmented with computer vision.

Users consider computer vision applications as technological, and therefore they will scrutinize it more than other additive technologies that offer incremental improvements. Unlike the task of making a voice call, which users already consider routine, they may find the interaction within computer vision application foreign and consequently difficult. Designers should pay attention to the following items:

- *Reliability:* The application should consistently produce accurate results as advertised. That is, if the application is supposed to detect a certain object, it should be able to do so all the time.
- *Responsiveness:* The application should complete its tasks within a reasonable amount of time. For example, users expect to take snapshots every few seconds.

FIFTH THIRD EX1062
PART 2 of 3
Page 301 of 379

Extra delay introduced with computer vision algorithms would be considered an annoyance.

- *Intuitiveness:* The application should present an interaction that is simple and natural. That is, the computer vision algorithm should not impose requirements, such as body gestures, that are unnatural or produce unnecessary fatigue.

Furthermore, users will have an innate desire to maintain privacy and security of their information. When users realize that computer vision applications have the ability to see and understand images captured from their camera, there is an immediate negative reaction towards the product feature. For safety- and security-related applications, such as fingerprint detection and face detection, users may relax their concerns. However, for other tasks, designers will have to face the challenge of adverse user attitudes.

Usability studies to explore the user interaction would be necessary to tune the application such that it becomes easier to use. "Once a technology becomes commonplace, people tend not to think of it as technological" [41]. At that point in time, user perception and attitudes of the usefulness of computer vision applications would have changed.

11.4.2 Measurability and Standardization

There is a lack of methods to measure the general effectiveness of computer vision implementations. Comparisons should include metrics that are quantitative and objective to produce results that are unbiased and repeatable.

There are existing standards to compare algorithms within certain applications. In face recognition systems, for example, there are ratings for successfully identifying a face (recognition rate) as well as incorrectly matching a face (false positives and false negatives). There is a database of facial images to standardize the evaluation system [42]. There is other existing research that attempts to standardize the performance of computer vision algorithms (i.e., how well it performs). In [43], testing procedures, metrics, and data are provided to standardize algorithm testing methodology. Furthermore, insights into algorithm design and its affect of implementation are provided.

What is needed are application benchmarks that can be used to determine the performance of computing platforms (i.e., how well it runs). Similar to MediaBench [44] and MineBench [45], these benchmarks would be useful to compare different embedded hardware solutions for computer vision. The suite should be constructed from a unique set of algorithms with prioritization based on how often that algorithm is used.

Discussions on measurability will eventually lead to discussions on standardization. There is a basic need to standardize the algorithm and application interface so that designers can easily integrate a number of different algorithmic parts to form a processing chain. The interface can be as simple as a parameter list for a C-function

FIFTH THIRD EX1062
PART 2 of 3
Page 302 of 379

call, or an elaborate protocol definition to define detectable objects. Open Source effort such as OpenCV [46-48] is a good example that would lead to standardization. This standardization effort may follow precedence from SMIA (standard mobile imaging architecture) [49,50] where there is a joint effort among 500 organizations to standardize the components in imaging modules, starting from sensors to housing and optics.

11.4.3 Business Models

It is important for designers of computer vision applications on mobile handsets to consider innovative business models when introducing the technology. As presented earlier in this chapter, there is a number of technical challenges that need to be resolved before productization. Even though computer vision can be a major technology disrupter [51] or a killer application to today's applications, business leaders may still find it risky. Technical issues must be resolved before they would consider it feasible for integration into a mobile handset. There may not be enough incentives to introduce computer vision applications in light of the technical challenges.

Designers often conceive new applications for technology's sake. In these applications, computer vision is only a fancy feature among many other gadgets or faddish technology of the time. For example, a gaming application using computer vision for tracking is nice, but is it absolutely necessary, especially with more accurate sensors? In most likelihood, users would not find the computer vision solution compelling because of user perception issues, described earlier in this chapter, and "feature fatigue" [17].

To lower such barriers, computer vision designers may want to avoid gadget-centric applications. Designers should consider a holistic approach in developing the application by including other components in the mobile handset and network. Computer vision algorithms can interact with the audio system, GPS navigation, web browsers, network servers, advertisement, and so on. Only with such integration with existing mobile handset infrastructure can designers realize the potential of computer vision. Only then can designers consider proper business models that can justify for computer vision applications' place in a mobile handset.

11.5 Future Direction

While this chapter has proposed several aspects of measuring effectiveness of a computer vision implementation with metrics such as reliability, responsiveness, and intuitiveness, this list is not complete and it lacks a proper rating scale. Designers of computer vision algorithms should consider the psychological aspects that influence the acceptance of the technology. With understanding of how users effectively use computer vision features in a mobile handset and what motivates them to

FIFTH THIRD EX1062
PART 2 of 3
Page 303 of 379

continue using the feature, designers can make inroads into having the technology as a commonplace feature set.

Furthermore, we should consider the sociological impact of pervasive computer vision technology in our everyday lives. This statement is not necessarily a call to monitor and examine every aspect of the technology in our society. Instead, it is an opinion for designers to consider computer vision applications that have great social impact. Technology such as computer vision can be applied to improve daily lives by making routine tasks faster and safer. We should seek to utilize mobile technology to improve the way to communicate and educate [52] ourselves.

This chapter started with the notion that the design of computer vision applications is complex and requires a team of experts with multidisciplinary skills. A range of issues, both technical and less technical, are then discussed along with insights into the implications of the computer vision algorithm operation. It would be evident that the mobile environment offers even more restrictive and unique challenges. Design teams would expand to consider usability and business aspects that would make a successful product deployment.

Despite these issues, some discussed here in this chapter and undoubtedly new ones to come, the prospects of mobile computer vision applications are excellent. Challenges can be overcome with properly directed research, and applications would arrive as momentum builds with this technology.

References

1. Brian O'Rourke, "CCDs & CMOS: Zooming in on the image sensor market," In-Stat Report IN030702MI, September 2003.
2. D. S. Wills, J. M. Baker, Jr., H. H. Cat, S. M. Chai, L. Codrescu, J. Cruz-Rivera, J. C. Eble, A. Gentile, M. A. Hopper, W. S. Lacy, A. López-Lagunas, P. May, S. Smith, and T. Taha, "Processing architecture for smart pixel systems," *IEEE J. Select Topics Quantum Electron,* v. 2, no 1, pp. 24-34, 1996.
3. Wayne Wolf, Burak Ozer, Tiehan Lv, "Smart cameras as embedded systems," *IEEE Computer,* September 2002, pp. 48-53
4. J. Adams, K. Parulski, and K. Spaulding, "Color processing in digital cameras," *IEEE Micro,* no. 18, pp. 20-30, 1998.
5. Andrew Wilson, "Understanding camera performance specs," *Vision Systems Design,* vol 12, no 7, July 2007, pp. 39-45.
6. Gregory K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM,* v. 34, no. 4, April 1991, pp.30-44.
7. Didier Le Gall, "MPEG: a video compression standard for multimedia applications," *Communications of the ACM,* Special issue on digital multimedia systems, v. 34, no. 4, April 1991, pp. 46-58.
8. Vasudev Bhaskaran, Konstantinos Konstantinides, *Image and Video Compression Standards,* 2nd edition, Kluwer Academic Press, 1997.
9. Xi-Ping Luo, Jun Li, Li-Xin Zhen, "Design and implementation of a card reader based on built-in camera," *Proceedings of the 17th International Conference on Pattern Recognition,* v. 1, 23-26 Aug. 2004, pp. 417-420.
10. J. Coughlan, R. Manduchi, "Color targets: fiducials to help visually impaired people find their way by camera phone," *EURASIP Journal on Image and Video Processing,* special issue on image and video processing for disability, v. 2007, article ID 96357, 2007.

FIFTH THIRD EX1062
PART 2 of 3
Page 304 of 379

11. Scalado AB, Lund, Sweden, “Scalado heralds the dawn of a ’new age’ for mobile imaging at the Mobile World Congress in Barcelona,” press release, www.scalado.com/m4n.
12. Eyal de Lara, Maria Ebling, “New products: motion-sensing cell phones,” *IEEE Pervasive Computing,* v 6, no 3, July-Sept. 2007, pp.15-17.
13. M.Sohn, G. Lee, “ISeeU: Camera-based user interface for a handheld computer,” *Mobile-HCI’05,* Sept 2005, pp. 299-302.
14. Sony Computer Entertainment, Inc., Sony Eye Toy, www.eyetoy.com.
15. Kris Graft, “Analysis: history of cell-phone gaming,” *Business Week,* January 22, 2006.
16. Y. Cheng, M.W. Maimone, L. Matthies, “Visual odometry on the Mars exploration rovers – a tool to ensure accurate driving and science imaging,” *IEEE Robotics & Automation Magazine,* v. 13, no. 2, June 2006, pp. 54-62.
17. Roland T. Rust, Debora V. Thompson, RebeccaW. Hamilton, “Defeating feature fatigue,” *Harvard Business Review,* Feb 1, 2006.
18. D. Talla, J. Gobton, “Using DaVinci technology for digital video devices,” *Computer,* v. 40, no.10, Oct. 2007, pp. 53-61.
19. Max Baron, “Freescale’s MXC voted best: the crown goes to Freescale’s MXC91321 chip,” *Microprocessor Report,* January 30, 2006, pp. 1-3.
20. Tomas Akenine-Müller, Jacob Strüm, “Graphics for the masses: a hardware rasterization architecture for mobile phones,” *ACM Transactions on Graphics (TOG),* v. 22, no 3, July 2003, pp. 801-808.
21. Pei Zheng, Lionel Ni, Lionel M. Ni, *Smart Phone and Next-Generation Mobile Computing,* Elsevier Science & Technology Books, December 2005.
22. Alan Zeichick, “Look Ma, no wires,” *NetNews,* v. 11, no. 4, December 2007, pp. 5-8.
23. Richard Harrison, Mark Shackman, *Symbian OS C++ for Mobile Phones,* Symbian Press, Wiley, 2007.
24. Tommi Mikkonen, *Programming Mobile Devices: An Introduction for Practitioners,* Wiley, 2007.
25. J. Owens et al., “A survey of general-purpose computation on graphics hardware,” *Proc. Eurographics,* 2005, pp. 21-51.
26. S. M. Chai, et al., “Streaming processors for next-generation mobile imaging applications,” *IEEE Communications Magazine,* Circuits for Communication Series, vol 43, no 12, Dec 2005, pp. 81-89.
27. M. Cummings, S.Haruyama, “FPGA in the software radio,” *IEEE Communications,* v. 37, no. 2, Feb 1999, pp. 108-112.
28. T. Tuan, S. Kao, A. Rahman, S. Das, S. Trimberger, “A 90-nm low-power FPGA for battery-powered applications,” *Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field-Programmable Gate Arrays,* Monterey, California, 2006, pp. 3-11.
29. A. López-Lagunas, S. M. Chai, “Memory bandwidth optimization through stream descriptors,” *ACM SIGARCH Computer Architecture Newsletter,* vol 34, no 1, pp. 57-64, March 2006.
30. S. Palacharla, R.E. Kessler, “Evaluating stream buffers as a secondary cache replacement,” *Proceedings of the 21st Annual International Symposium on Computer Architecture,* pp. 24-33, April 1994.
31. S. A. McKee, et. al., “Dynamic access ordering for streamed computations,” *IEEE Transactions on Computers,* Vol. 49, No. 11, November 2000.
32. L. Zhang, Z. Fang, M. Parker, B. K. Mathew, L. Schaelicke, J. B. Carter, W. C. Hsieh, S. A. McKee, “The impulse memory controller,” *IEEE Transactions on Computers,* pp. 1117-1132, Nov 2001.
33. A. Bellaouar, M. I. Elmasry, *Low-Power Digital VLSI Design: Circuits and Systems,* Springer, June 30, 1995.
34. W. Bidermann, A. El Gamal, S. Ewedemi, J. Reyneri, H. Tian, D. Wile, D. Yang, “A 0.18 /spl mu/m high dynamic range NTSC/PAL imaging system-on-chip with embedded DRAM frame buffer,” *IEEE International Solid-State Circuits Conference,* v.1, 2003, pp. 212-488.
35. S. B. Gokturk, H. Yalcin, C. Bamji, “A time-of-flight depth sensor - system description, issues and solutions,” *Computer Vision and Pattern Recognition Workshop,* June 2004, p. 35.

FIFTH THIRD EX1062
PART 2 of 3
Page 305 of 379

36. Eugene Hecht. *Optics* (4th ed.). Pearson Education. 2001.
37. N. Paragios, Y. Chen, and O. Faugeras, eds., *The Handbook of Mathematical Models in Computer Vision*, Springer, 2005.
38. B. Berge, "Liquid lens technology: principle of electrowetting based lenses and applications to imaging," *Proc. IEEE International Conference on Micro Electro Mechanical Systems*, 2005.
39. E. J. Tremblay, R. A. Stack, R. L. Morrison, and J. E. Ford, "Ultrathin cameras using annular folded optics," *Applied Optics*, Vol. 46, Issue 4, pp. 463-471.
40. Martin Buehler, Karl Iagnemma, and Sanjiv Singh, *The 2005 DARPA Grand Challenge: The Great Robot Race*, Springer, 2007.
41. C. Lankshear, I. Snyder, *Teachers and Technoliteracy: Managing Literacy, Technology and Learning in Schools*, St. Leonards, NSW, Australia: Allen & Unwin, 2000.
42. P. J. Phillips, M. Hyeonjoon, S.A. Rizvi, and P.J. Rauss, "The FERET evaluation methodology for face-recognition algorithms" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, no. 10, Oct. 2000, pp. 1090-1104.
43. P. Courtney, N. A. Thacker, "Performance Characterization in Computer Vision." In *Imaging and Vision Systems*, Jacques Blanc-Talon and Dan Popescu (Eds.), NOVA Science Books, 2001.
44. Chunho Lee, Miodrag Potkonjak, William H. Mangione-Smith, "MediaBench: a tool for evaluating and synthesizing multimedia and communicatons systems," *Proceedings of the 30th annual ACM/IEEE International Symposium on Microarchitecture*, 1997, pp. 330-335.
45. R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, A. Choudhary, "MineBench: A benchmark suite for data mining workloads," *2006 IEEE International Symposium on Workload Characterization*, Oct. 2006, pp. 182-188.
46. OpenCV. http://www.intel.com/research/mrl/research/opencv/
47. Gary Bradski, Adrian Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, Inc., 2008.
48. Petri Honkamaa, Jani Jäppinen, Charles Woodward, "A lightweight approach for augmented reality on camera phones using 2D images to simulate 3D," *Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia*, vol. 284, Oulu, Finland, 2007, pp. 155-159.
49. SMIA: Standard Mobile Imaging Architecture, http://www.smia-forum.org.
50. Lee Nelson, "Solving the Problems of Mobile Imaging," *Advanced Imaging*, vol 22, no 4, April 2007, pp. 10-13.
51. Clayton M. Christensen, *The Innovator's Dilemma: The Revolutionary Book that Will Change the Way You Do Business*, Collins, 2003.
52. David Metcalf, *M-Learning: Mobile E-Learning*, HRD Press, Inc., January 2006.

FIFTH THIRD EX1062
PART 2 of 3
Page 306 of 379

APPENDIX
HARRISON19

THESEUS

Finnish | Swedish | English » Login

Search SEARCH Application instructions

Search for Theseus This collection

Universities of applied sciences / Tampere University of Applied Sciences / Theses (Open collection) / Show reference

Show brief description information

UIQ application that utilizes the touchscreen

Liimatainen, Tuomas (2009)

dc.contributor.author	Liimatainen, Tuomas
dc.date.accessioned	2010-03-08T09:01:39Z
dc.date.available	2010-03-08T09:01:39Z
dc.date.issued	2009
dc.identifier.uri	URN:NBN:fi:amk-201003064988
dc.identifier.uri	https://www.theseus.fi/handle/10024/10201
dc.language.iso	fin
dc.publisher	Tampere University of Applied Sciences
dc.rights	All rights reserved
dc.subject	UIQ
dc.subject	Symbian
dc.title	UIQ application that utilizes the touchscreen
dc.type.ontasot	fi=AMK-opinnäytetyö|sv=YH-examensarbete|en=Bachelor's thesis|
dc.identifier.dscollection	10024/6717
dc.organization	Tampere University of Applied Sciences
dc.subject.ysa	operating systems
dc.subject.ysa	mobile phones
dc.subject.ysa	touchscreen
dc.subject.ysa	user interfaces
dc.subject.ysa	Symbian OS
dc.contributor.organization	Tampere University of Applied Sciences
dc.subject.degreeprogram	fi=Information and Communications Technology|sv=Informations- och kommunikationsteknik|en=Information and Communications Technology|
dc.subject.discipline	Information Technology Degree Program

Files



Name: Liimatainen.Tuomas.pdf Open file
Size: 1.198MB
Format: PDF

BROWSE THE COLLECTION

» Titles
» Authors
» Publication times
» Educational fields
» Keywords
» Latest
» Holding

The reference belongs to the collections:

- Theses (Open collection)

Show brief description information

THESEUS | Ammattikorkeakoulujen opinnäytetyöt ja julkaisut

Theses and publications from universities of applied sciences
Contact | About usage rights | Privacy policy | Accessibility statement

TAMPEREEN AMMATTIKORKEAKOULU OPINNÄYTETYÖ
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Opinnäytetyö

Tuomas Liimatainen

KOSKETUSNÄYTTÖÄ HYÖDYNTÄVÄ UIQ-SOVELLUS

Työn ohjaaja Lehtori Jari Mikkolainen
Työn teettäjä Sasken Finland Oy, valvojana diplomi-insinööri Jani Turpeinen
Tampere 2009

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Liimatainen, Tuomas
Kosketusnäyttöä hyödyntävä UIQ-sovellus
Tutkintotyö 44 sivua
Työn ohjaaja Lehtori Jari Mikkolainen
Työn teettäjä Sasken Finland Oy, valvojana diplomi-insinööri Jani Turpeinen
Toukokuu 2009
Hakusanat UIQ, Symbian, käyttöliittymä, kosketusnäyttö

TIIVISTELMÄ

Symbian on älypuhelimissa yleisimmin käytetty käyttöjärjestelmä, johon perustuvista matkapuhelinalustoista yleisimpiä ovat S60- ja UIQ-alustat. S60-alusta on selvästi yleisin Symbianiin perustuva alusta, jota käytetään Nokian älypuhelimissa. Koska kosketusnäytön hyödyntäminen on viime aikoina noussut tärkeäksi kilpailutekijäksi eri laitevalmistajien ja matkapuhelinalustojen välillä, julkaisi Nokia ensimmäisen S60-alustalla toimivan ja kosketusnäyttöä hyödyntävän matkapuhelimen vuonna 2008. UIQ-alusta on alusta alkaen suunniteltu kosketusnäyttöä ajatellen ja esimerkiksi Sony Ericsson on tuonut markkinoille kosketusnäyttöä hyödyntäviä UIQ-puhelimia jo usean vuoden ajan.

Tämän työn tarkoituksena oli perehtyä S60- ja UIQ-alustoihin ja erityisesti kosketusnäytön hyödyntämiseen näillä alustoilla. Alustoihin perehtymisen lisäksi, tehtävänä oli siirtää Sasken Finland Oy:ssä S60-alustalle kehitetty testausympäristö UIQ-alustalle siten, että kosketusnäytön ominaisuuksia hyödynnetään ja että käyttöliittymästä tulee entistäkin kätevämpi. Sovelluksen käyttöliittymä piti toteuttaa siten, että toimintojen suorittaminen on mahdollisimman helppoa ja selkeää kosketusnäytön avulla.

Testausympäristön siirtäminen S60-alustalta UIQ-alustalle saatiin toteutettua suunnitellusti. Käyttöliittymää on paranneltu ja käyttö kosketusnäytön avulla on helppoa ja selkeää.

TAMPERE UNIVERSITY OF APPLIED SCIENCES

Computer Systems Engineering
Software Engineering
Liimatainen, Tuomas UIQ-application utilizing touch screen
Engineering Thesis 44 pages
Thesis Supervisor Senior Lecturer Jari Mikkolainen
Commissioning Company Sasken Finland Ltd. Supervisor M.Sc. Jani Turpeinen
May 2009
Keywords UIQ, Symbian, user interface, touch screen

ABSTRACT

Symbian is the most wide spread smart phone operating system and S60- and UIQ-platforms are the most common platforms that are based on it. Because touch screen support has recently become very important factor in competition between different mobile phone manufacturers and –platforms, Nokia released it's first S60-platform based phone with touch screen support in 2008. UIQ-platform has been designed to support touch screens from the beginning and Sony Ericsson for example has been selling UIQ-phones with touch screen support for many years.

The purpose of this work was to familiarize oneself with S60- and UIQ-platforms and especially with touch screen usage on them. In addition to getting to know the platforms, a test environment, developed in Sasken Finland Ltd., had to be ported from S60- to UIQ-platform. Touch screen utilization had to be taken into account when porting the application. The application had to have a user interface that is easy to use and takes advantage of the possibilities provided by touch screen.

Porting of the test environment was done as planned. The user interface has been improved and usage of the application via touch screen is easy and clear.

SISÄLLYSLUETTELO

TIIVISTELMÄ
ABSTRACT

1 JOHDANTO 6
2 Symbian 6
2.1 Symbianin sovelluskehys 6
2.2 Symbian-käyttöjärjestelmään perustuvat matkapuhelinalustat 8
2.3 Symbianin tuki kosketusnäytölle 10
2.3.1 Näppäintapahtumien käsittely 11
2.3.2 Kosketustapahtumien käsittely 12
2.3.2.1 Kosketustapahtuman kulku ja käsittely yleisesti 12
2.3.2.2 Eri kosketustapahtumatyypit ja niiden käsittely 13
3 S60-alusta 14
3.1 S60-sovellusten tyyli ja käyttölogiikka 14
3.2 Kosketusnäyttö S60-alustalla 16
3.2.1 Yleistä kosketusnäytöstä S60-alustalla 16
3.2.2 S60:n 5.0-versiossa esitellyt käyttöliittymärajapinnat 17
4 UIQ-alusta 19
4.1 UIQ-sovellusten tyyli ja käyttölogiikka 19
4.1.1 Yleistä 19
4.1.2 UIQ:n eri UI-konfiguraatiot 20
4.1.2.1 Toimintonäppäintyylinen käyttöliittymä 21
4.1.2.2 Kynätyylinen käyttöliittymä 23
4.1.2.3 Eri UI-konfiguraatioihin varautuminen 25
5 Wireless Test Framework 27
5.1 Yleistä testikehyksestä 27
5.2 Käyttöliittymä 28
5.2.1 Rakenne 28
5.2.2 Näkymät sekä toiminnallisuudet 30
5.2.2.1 Tapausnäkymä 31
5.2.2.2 Moduulinäkymä 33
5.2.2.3 Ajonäkymä 35
5.2.3 Käyttöliittymän parannukset 37
5.2.4 Toiminta eri UI-konfiguraatioilla 38
5.3 Ongelmakohdat 39
5.3.1 Käytöstä poistettu dialogi 40
5.3.2 UIQ yhtiön konkurssi 40
5.4 Työn onnistuminen 41
5.5 Jatkokehitysajatuksia 41

LÄHDELUETTELO

LYHENTEIDEN SELITYKSET

API	Application Programming Interface on ohjelmointirajapinta, jonka kautta eri ohjelmat tai muut yksiköt voivat keskustella keskenään.
Avkon	S60-alustan käyttöliittymäkirjastot kuuluvat Avkon-kehykseen, joka laajentaa UIKON-kirjastoa. Avkon sisältää käyttöliittymään liittyviä komponentteja ja kontrolleja, joista S60-alustan käyttöliittymä muodostuu.
DLL	Dynamically Linked Library on jaettu kirjasto, josta voidaan ajon aikana ladata funktioita suoritettavaksi.
Eikon	UIKON-kehys oli ennen nimeltään Eikon.
EPOC	Symbian-käyttöjärjestelmä oli ennen nimeltään EPOC.
FEP	Front End Processor on yksikkö, joka esikäsittelee dataa ennen sen siirtämistä eteenpäin.
MVC	Model View Controller -arkkitehtuuri on graafisissa sovelluksissa yleisesti käytetty arkkitehtuurityyli, jossa sovellus jaetaan kolmeen osaan: malliin, näkymään ja kontrolleriin.
Qikon	UIQ-alustan käyttöliittymäkirjastot kuuluvat Qikon kehykseen, joka laajentaa UIKON-kirjastoa. Qikon sisältää käyttöliittymään liittyviä komponentteja ja kontrolleja, joista UIQ-alustan käyttöliittymä muodostuu.
SDK	Software Development Kit on kokoelma kehitystyökaluja ja kirjastoja, joiden avulla voidaan kehittää sovelluksia jollekin tietylle alustalle.
Tactile Feedback	Tactile Feedback on tekniikka, jolla käyttäjälle annetaan palautetta kosketusnäytön painalluksista S60-alustalla.
UI	User Interface, käyttöliittymä.
UIKON	Symbian-käyttöjärjestelmän käyttöliittymän kirjastot kuuluvat UIKON-kehykseen, joka on perusta graafisille Symbian-sovelluksille.
WTF	Wireless Test Framework on Sasken Finland Oy:ssä kehitetty testausympäristö.

1 JOHDANTO

Kosketusnäytöt ovat viime aikoina yleistyneet huikeaa vauhtia matkapuhelinmarkkinoilla. Suurimman huomion kosketusnäyttöpuhelimista on saanut Applen julkaisema iPhone, vaikka esimerkiksi Sony Ericsson on tehnyt ja myynyt kosketusnäyttöä hyödyntäviä puhelimia jo useita vuosia. Myös Googlen kehittämä Android-käyttöjärjestelmä hyödyntää kosketusnäyttöjä.

Tämän työn tarkoitus on perehtyä Symbian-käyttöjärjestelmään perustuviin UIQ- ja S60-alustoihin ja kosketusnäytön hyödyntämiseen niissä. Työn teoriaosuudessa perehdytään Symbianin sovelluskehykseen, sen tukeen kosketusnäytölle ja siihen perustuviin matkapuhelinalustoihin. Työn ohjelmointivaiheessa toteutetaan testikehyksen siirtäminen UIQ-alustalle keskittyen toimivaan käyttöliittymään, joka hyödyntää kosketusnäyttöä.

2 SYMBIAN

Tässä luvussa tutustutaan Symbian-sovellusten rakenteeseen yleisesti, sekä annetaan yleiskuvaus kahdesta suurimmasta Symbianiin perustuvasta matkapuhelinalustasta. Lisäksi tutustutaan Symbianin tukeen kosketusnäytölle.

2.1 Symbianin sovelluskehys

Käyttäjäystävällisten matkapuhelinsovellusten tärkein ominaisuus on toimiva, graafinen käyttöliittymä. Symbian-alustalla graafisten sovellusten tukemisessa keskeisessä asemassa on sovelluskehyksen alijärjestelmä UIKON.

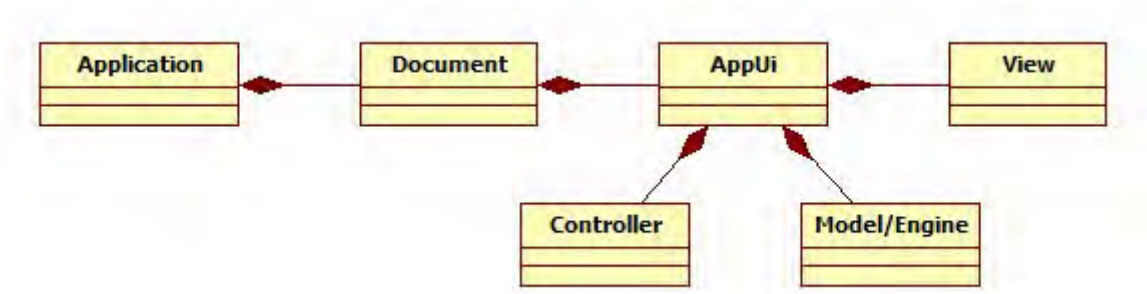
UIKON tarjoaa kehyksen, jonka avulla voidaan luoda graafisen käyttöliittymän sisältäviä sovelluksia. Kehyksen arkkitehtuuri kannustaa käyttämään MVC-mallia, jossa sovellus jaetaan erillisiin loogisiin osiin, joilla on sovelluksessa tietty tehtävä. MVC-mallin mukaiset tehtävät ovat malli (engl. model), näkymä (engl. view) sekä kontrolleri (engl. controller). UIKON tarjoaa perusluokat graafisten Symbian

sovellusten luontiin. Nämä luokat noudattavat MVC-mallia ja ovat listattuna taulukossa 1. /5, s. 312/

Luokka	Kuvaus
Application - CEikApplication	Sovelluksen käynnistyspiste, joka luo document-luokan ilmentymän, määrittää sovelluksen ominaisuuksia sekä tarjoaa rajapinnan sovelluksen resurssitiedostoon.
Document - CEikDocument	Toimii MVC-mallissa mallina. Sovelluksen ollessa tiedostopohjainen, document-olio on vastuussa tiedon tallentamisesta ja lataamisesta. Tämän luokan ilmentymä luo Application UI-luokan.
Application UI - CEikAppUi	Toimii MVC-mallissa kontrollerina. Vaikka tämä luokka on nimeltään Application UI, se on täysin näkymätön käyttäjälle. Tämän luokan ilmentymä luo näkymäluokan tai -luokkia sekä käsittelee usein käyttäjän komentoja, kuten valikoista tehtyjä valintoja.
View – luokka, joka periytyy CCoeControl:sta	Toimii MVC-mallissa näkymänä. Tämän luokan tarkoitus on näyttää sovelluksen tila käyttäjälle ja antaa käyttäjän kommunikoida sovelluksen kanssa. Myös näkymäluokka voi käsitellä käyttäjän komentoja, esimerkiksi näppäinpainalluksia.

Taulukko 1 *UIKON-kehyksen perusluokat graafisen sovelluksen luontiin* /5, s. 312/

Kaikkien graafisten Symbian-sovellusten tulee sisältää taulukossa 1 esiteltyjen luokkien toteutukset. Vaikka UIKONin tarjoamat luokat sinällään noudattavat MVC-mallia, käytetään Symbian-sovelluksissa usein kuitenkin erillisiä luokkia mallin toteuttamiseen. Kuvassa 1 on esitetty tyypillisen sovelluksen yleisen tason luokkakaavio.



Kuva 1 *MVC-mallisten sovellusten yleisluokkakaavio*

MVC-arkkitehtuurissa näkymä, kontrolleri ja malli ovat erillisiä komponentteja, joiden välillä kommunikointi tapahtuu tiettyjen sääntöjen mukaan. Komponentit ovat erillisiä yksiköitä, jolloin jonkun yksikön toteutusta voidaan muuttaa, ilman että se vaikuttaa toisiin yksiköihin.

2.2 Symbian-käyttöjärjestelmään perustuvat matkapuhelinalustat

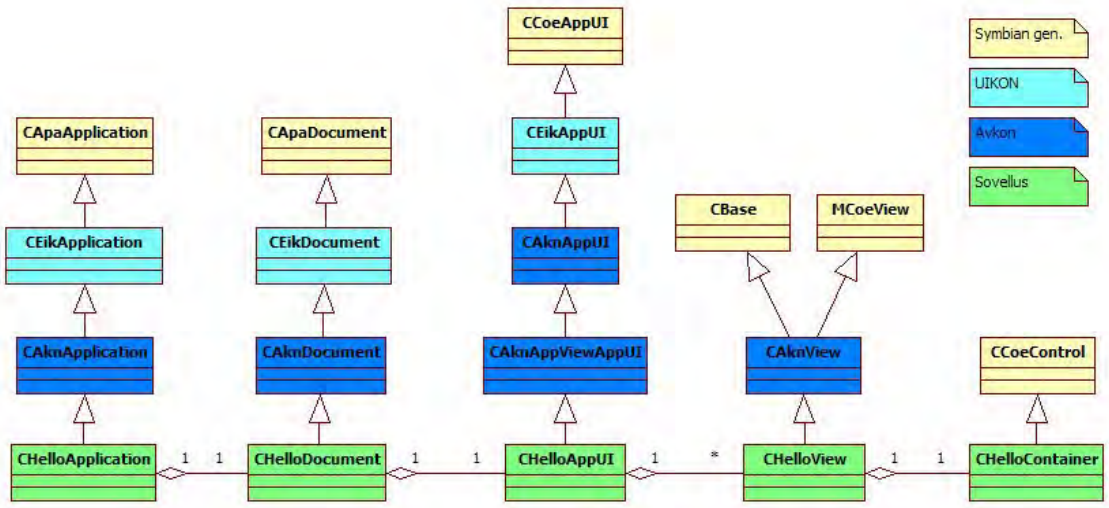
Symbian on matkapuhelimissa käytettävä käyttöjärjestelmä, johon perustuen on tehty useita eri matkapuhelinalustoja. Euroopassa tunnetuimpia alustoja ovat Nokian kehittämä ja käyttämä S60 ja esimerkiksi Sony Ericssonin ja Motorolan käyttämä UIQ. Kolmas tunnettu Symbianiin perustuva alusta on MOAP. Tätä alustaa käyttää NTT DOCOMO ja se on Aasiassa tunnetumpi kuin UIQ. Tässä opinnäytetyössä keskitytään S60- ja UIQ-alustoihin, eikä MOAP-alustaa käsitellä.

S60- ja UIQ-alustat perustuvat Symbianiin ja käyttävät täten hyväkseen kappaleessa 2.1 esitettyä UIKON-kehystä ja sen luokkia. Eri alustat laajentavat Symbianin tarjoamaa kehystä omilla kirjastoillaan, joiden avulla käyttöliittymä saadaan muokattua kohdelaitteille sopivaksi. Kohdelaitteissa on eroja esimerkiksi näytön resoluution ja orientaation sekä eri vuorovaikutustapojen suhteen. Laitetta voidaan käyttää esimerkiksi erilaisilla näppäimistöillä tai kosketusnäytön avulla. /5, s. 313/

Vaikka eri alustat on suunniteltu hieman eri tavoilla ja erilaisin tavoittein, on pohjalla kuitenkin yhteinen käyttöjärjestelmä. Yhteisen käyttöjärjestelmän vuoksi, alustojen ohjelmointirajapinnat ovat usein saman kaltaisia. S60-alustan lisäämän

kirjaston nimi on Avkon ja UIQ:n Qikon. Käytännössä suurin osa näiden alustojen luokista periytyy Symbianin luokista, ja niihin on vain lisätty eri alustoilla tarvittavia ominaisuuksia. /5, s. 313/

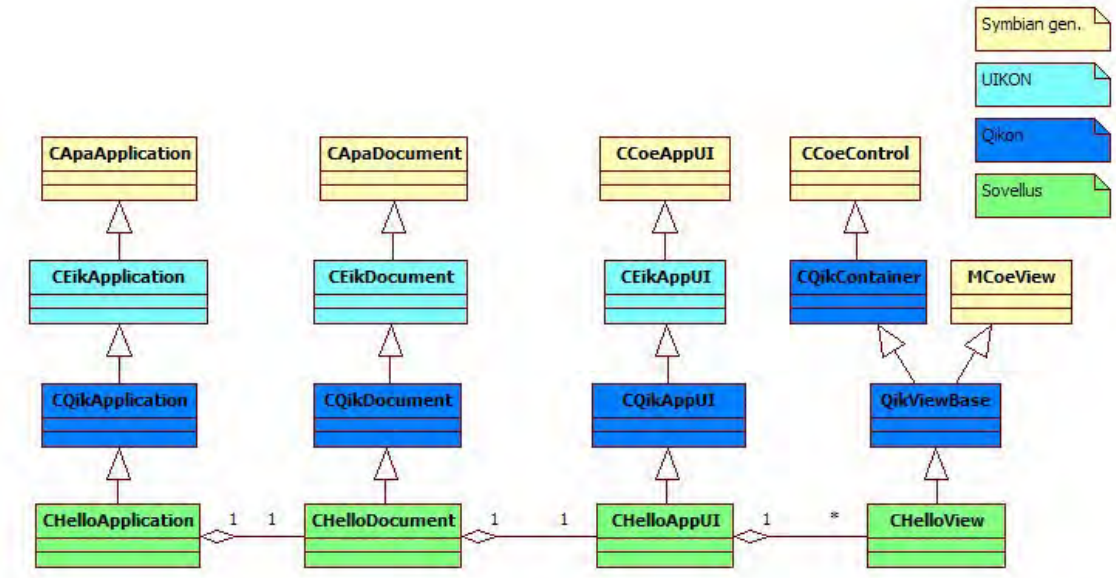
Koska molemmat alustat perustuvat samaan käyttöjärjestelmään, on S60- ja UIQ-sovellusten rakenne lähes sama. Vaikka rakenne on saman kaltainen, saadaan Avkon- ja Qikon-alustojen avulla sovelluksista ja käyttöliittymistä erilaiset. Molemmat alustat tukevat useita erilaisia käyttöliittymiä, kuten erilaisia näytön resoluutioita ja orientaatioita. S60-sovelluksen rakenne sekä luokkien periytyminen on esitetty kuvassa 2.



Kuva 2 *S60-sovelluksen luokkien periytyminen*

Yllä olevasta kuvasta nähdään, että Avkon-alustan luokat voidaan helposti tunnistaa Akn-etuliitteestä. Vastaavasti UIKON-kehyksen luokkien alussa käytetään etuliitettä Eik. Eik-etuliite johtuu siitä, että alunperin Symbian-käyttöjärjestelmän käyttöliittymän sovelluskehyksen nimi oli Eikon ja käyttöjärjestelmä itse tunnettiin nimellä EPOC. Myöhemmin Eikon-kehyksen nimi muutettiin UIKONiksi, mutta luokkien nimiä ei vaihdettu. Jos luokan nimessä ei ole Akn- eikä Eik-etuliitettä, on kyseessä geneerinen Symbian-luokka. Tämä ei koske kuvan alinta riviä, jonka luokat sovelluksen tekijä voi nimetä kuten haluaa. /5, s. 311-313/

Kuvasta nähdään myös, että S60-alustalla näkymäluokka ei periydy CCoeControl-luokasta, eli se ei itse ole kontrolli. Tämä on poikkeavaa UIQ:hun verrattuna, sillä UIQ:ssa näkymä periytyy suoraan CCoeControlista, joten se on itsekin kontrolli. Tämän ansiosta UIQ-sovelluksissa ei ole erillistä container-luokkaa. UIQ-sovelluksen rakenne sekä luokkien periytyminen on esitetty kuvassa 3.



Kuva 3 *UIQ- sovelluksen luokkien periytyminen*

Kuten S60:n, myös UIQ:n luokat on nimetty tiettyä etuliitettä käyttäen. UIQ-alustalla etuliite on Qik. Myöhemmissä kappaleissa tutustutaan hieman tarkemmin näihin alustoihin sekä erityisesti niiden käyttöliittymien tyyliin ja käyttölogiikkaan.

2.3 Symbianin tuki kosketusnäytölle

Eräs tämän opinnäytetyön lähtökohdista oli tutustua kosketusnäyttöä hyödyntävän sovelluksen tekoon UIQ-alustalla. Kun UIQ-alustaa kehitettiin, oli alusta asti selvä, että sen piti tukea kosketusnäyttöä. Kosketusnäytön hyödyntämisen mahdollistaa Symbianin tuki niille. Kosketusnäytön koskettaminen saa aikaan kosketustapahtuman (engl. pointer event), joka on osittain samanlainen kuin puhelimen fyysiseltä näppäimistöltä tuleva tapahtuma.

Kosketus- ja näppäintapahtumien käsittely poikkeaa selvästi toisistaan. Jotta näemme kosketusnäytön kautta tulevien tapahtumien käsittelyn erot perinteiseltä näppäimistöltä tulevien tapahtumien käsittelyyn, tutustumme seuraavissa kappaleissa molempiin tapahtumiin.

2.3.1 Näppäintapahtumien käsittely

Näppäintapahtuma voi tulla joko näppäimen painalluksesta tai front-end prosessorilta (FEP), joka on eräänlainen esiprosessori. FEP:tä käytetään esimerkiksi ennakoivan tekstinsyötön, käsialan tunnistuksen, äänen tunnistuksen sekä kosketusnäytöllä olevan virtuaalinäppäimistön tapahtumien käsittelyyn. Esiprosessori tuottaa näppäintapahtumia esimerkiksi virtuaalinäppäimistön tietyn näppäimen painalluksesta. Kun näppäintä painetaan, FEP käsittelee tapahtuman ja muuntaa sen sopivaksi näppäintapahtumaksi. Sovelluksen ei tarvitse tietää, tuleeko näppäintapahtuma fyysiseltä näppäimistöltä vai FEP:ltä, koska tapahtuman käsittely on molemmissa tapauksissa sama. /9; 5, s. 432-433/

Näppäintapahtumat käsitellään kontrollipinon (engl. control stack) OfferKeyEventL-funktiossa, joka kutsuu vastaavaa funktiota jokaiselle pinossa olevalle kontrollille. Niitä selataan, kunnes joku niistä käsittelee tapahtuman ja palauttaa tätä vastaavan arvon tai kaikki kontrollit on käyty läpi, eikä mikään kontrolli ole käsitellyt tapahtumaa. Jos mikään kontrolli ei käsitellyt tapahtumaa, tulee se konrollipinon omistajan HandleKeyEventL-funktioon käsiteltäväksi. /5, s. 434/

Sovelluksen oman logiikan pitää hoitaa näppäintapahtumien käsittely. Esimerkiksi yhdistelmäkontrollit (engl. compound control) voivat itse käsitellä tapahtuman, tarjota tapahtumaa jollekin osakontrollille käsiteltäväksi tai olla käsittelemättä sitä ollenkaan. Sovelluksen sisäinen toteutus siis määrää, mikä kontrolli tapahtuman käsittelee ja mihin se vaikuttaa. Yksi esimerkki näppäintapahtumien käsittelystä voisi olla sellainen, jossa yhdistelmäkontrolli käsittelee nuolinäppäimet ja vaihtaa niiden avulla osakontrolliensa fokusta. Osakontrollit käsittelisivät muita näppäimiä tarpeensa mukaan. /5, s. 432, s. 434-437/

2.3.2 Kosketustapahtumien käsittely

Tässä kappaleessa tutustutaan kosketustapahtumien kulkuun järjestelmässä, niiden käsittelyyn, niiden eroihin näppäintapahtumiin verrattuna ja erilaisiin kosketustapahtumatyyppeihin.

2.3.2.1 Kosketustapahtuman kulku ja käsittely yleisesti

Kosketustapahtumat eroavat näppäintapahtumista siten, että ne kohdistuvat suoraan johonkin tiettyyn kontrolliin, eikä sovelluksen oman logiikan tarvitse hoitaa niiden välittämistä halutulle kontrollille. Ikkunapalvelin (engl. window server) assosioi tapahtuman tiettyyn, esillä olevaan ikkunaan, jonka alueella kosketus tapahtui, ja välittää tapahtuman ikkunaryhmän omistamalle sovellukselle. /5, s. 440-441/

Sovelluksen sisällä kosketustapahtumien käsittely etenee seuraavasti: Ensin tapahtuma välitetään Application UI-luokalle, jonka HandleWsEventL-funktiossa se tunnistetaan tiettyyn ikkunaan assosioiduksi kosketustapahtumaksi. Funktiosta tehdään kutsu ikkunan omistaman kontrollin ProcessPointerEventL-funktioon, josta kutsutaan HandlePointerEventL-funktiota. Tässä funktiossa voidaan tehdä halutut asiat kosketustapahtuman sattuessa. Jos kontrolli, jolle tapahtuma saapui, on yhdistelmäkontrolli, on sen oletustoteutus sellainen, että se käy läpi kaikki näkyvät osakontrollinsa ja välittää tapahtuman sille kontrollille, joka sijaitsee sillä kohtaa, jossa tapahtuma tapahtui. /5, s. 440-441/

Kosketustapahtumien käsittely on siinä mielessä samanlaista kuin näppäintapahtumien, että kontrolli voi olla käsittelemättä tapahtumaa tai se voi käsitellä tapahtuman ja tuottaa esimerkiksi komennon, joka käsitellään muualla. Toisaalta, kun kosketustapahtumat toimitetaan suoraan tietylle kontrollille, ei kosketustapahtumia normaalisti tarvitse välittää toisille kontrolleille, eikä tapahtumaa tarjota järjestelmän toimestakaan muille kontrolleille. /5, s. 441/

2.3.2.2 Eri kosketustapahtumatyypit ja niiden käsittely

Kosketustapahtumaa edustaa TPointerEvent-tyypin ilmentymä, joka on samankaltainen kuin näppäintapahtumaa edustava TKeyEvent. TPointerEvent sisältää tiedon siitä, oliko tapahtuman tyyppi kynän alas painaminen vai ylös nostaminen. Lisäksi saadaan tieto esimerkiksi tapahtuman toistosta ja sijainnista sekä siitä, onko Shift-, Ctrl- tai Alt-näppäin painettuna. Tapahtuman toistolla tarkoitetaan sitä, kun kynää pidetään painettuna samassa kohdassa pidemmän aikaa. /5, s. 439-440/

Kosketustapahtumien käsittelyssä pitää ottaa huomioon tapahtuman tyypin ja sijainnin lisäksi muitakin asioita. Tapahtumien käsittelyssä on huomioitava, että kynää voidaan siirtää näytön pinnalla, jolloin kynän nostamistapahtuma ei välttämättä tapahdu saman kontrollin kohdalla, kuin kynän laskemistapahtuma. Mahdollinen on myös tilanne, jossa ei haluta antaa kosketustapahtumia millekään muulle, kuin jollekin tietylle kontrollille. Näiden tilanteiden hallitsemiseksi Symbian tarjoaa seuraavat tekniikat: pointer grab ja pointer capture. /5, s. 440/

Pointer grab on tekniikka, jonka avulla varmistetaan, että kontrolli, jonka päällä tapahtuu kynän alas painaminen, saa käsiteltäväkseen myös seuraavat kosketustapahtumat, kunnes tulee seuraava kynän nostotapahtuma. Kynän nostotapahtuma lähetetään siis samalle kontrollille, vaikka kynä nostettaisiinkin ylös jonkun muun kontrollin kohdalla. Esimerkiksi näytöllä näkyvää nappia painettaessa piirretään nappi uudelleen, siten, että käyttäjä näkee sen olevan painettuna. Kun käyttäjä nostaa kynän näytön pinnalta, piirretään näppäin uudelleen ylääsentoon, ja jos kynä oli nostotapahtuman aikana saman näppäimen päällä, suoritetaan siihen liitetyt toiminnot. /5, s. 572-573/

Pointer grabin toimimiseksi pitää ikkunapalvelimen muistaa oikea ikkuna ja kontrolliympäristön pitää muistaa oikea kontrolli. Tämän ominaisuuden toteuttaa aiemmin mainittu ProcessPointerEventL-funktio, joka tekee myös muuta hyödyllistä esikäsittelyä kosketustapahtumille, kuten

- kynällä painaminen-tapahtuman hylkääminen näkymättömiltä kontrolleilta
- tapahtumien hylkääminen, kunnes tulee seuraava kynän nostamistapahtuma

- himmennettyjen kontrollien kosketustapahtumien raportointi ja hylkääminen
- kynällä painaminen-tapahtumien raportointi sellaisten kontrollien kohdalla, jotka eivät ole fokusoituja, mutta jotka ovat kykeneviä ottamaan fokuksen /5, s. 573, s. 441-442/

Pointer capture on tekniikka, jonka avulla voidaan estää kaikkia muita kontrolleja saamasta käsiteltäväkseen kynällä painaminen-tapahtumia. Tätä tekniikkaa hyödynnetään esimerkiksi dialogeissa, jolloin voidaan unohtaa kaikki dialogin ulkopuolella tapahtuneet kosketukset, sekä valikoissa, jolloin kosketus valikon ulkopuolella voidaan käsittää haluksi sulkea valikko. Kun valikko saa käsiteltäväkseen myös sen ulkopuolella tapahtuneen kosketuksen, se voi sulkea itsensä. /5, s. 440/

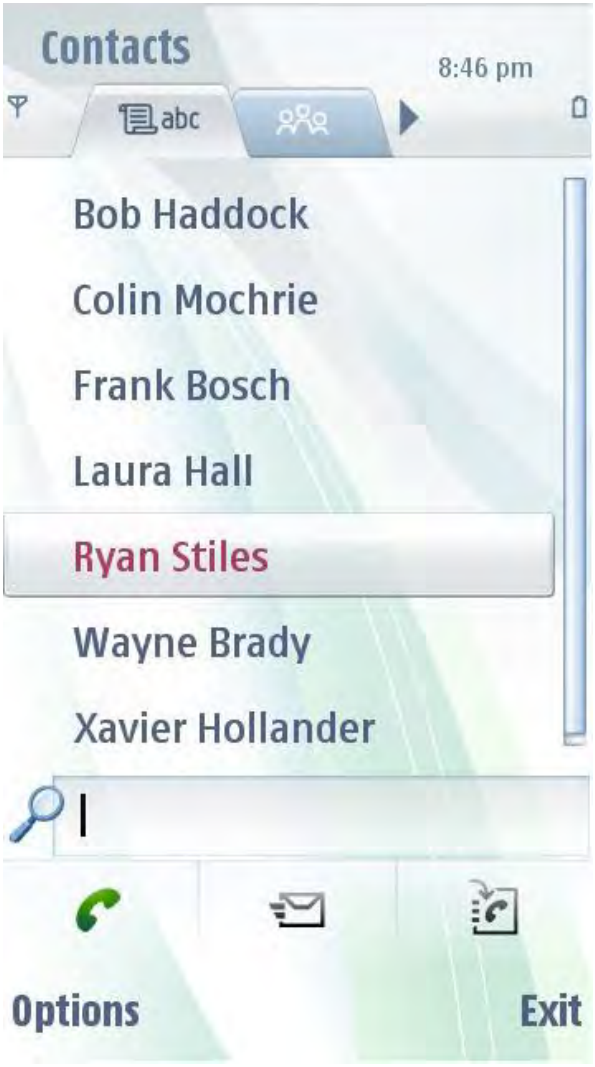
3 S60-ALUSTA

S60-alustan uusin versio, 5.0, julkistettiin syksyllä 2008. Uusin versio perustuu Symbianin versioon 9.4, joka toi ensimmäistä kertaa tuen kosketusnäytölle S60-alustalle. Tässä kappaleessa tutustutaan hieman tarkemmin S60-alustan sovellusten käyttöliittymään sekä alustan tukeen kosketusnäytölle.

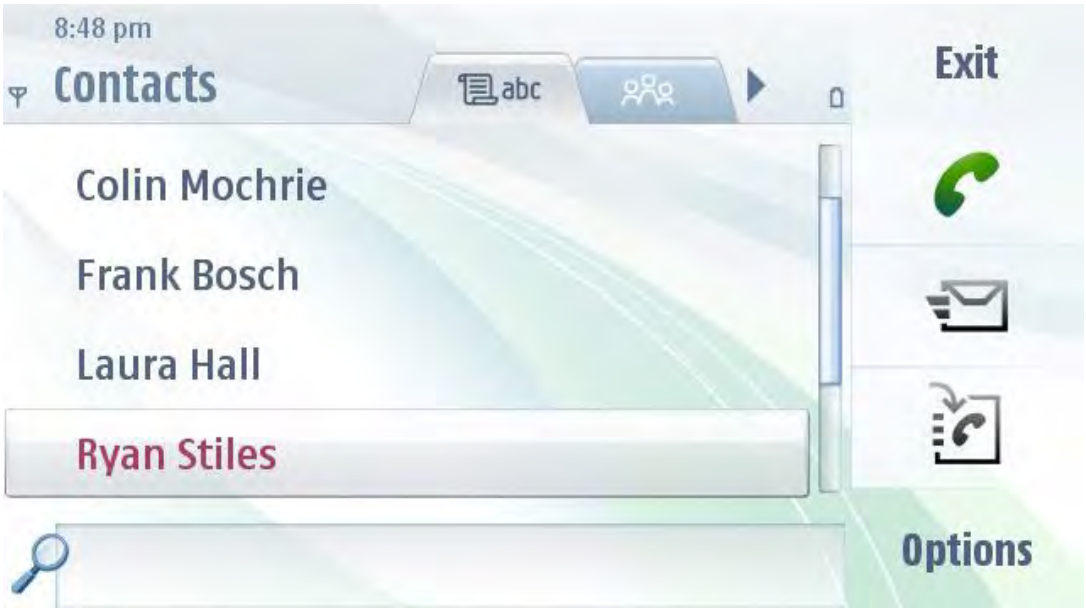
3.1 S60-sovellusten tyyli ja käyttölogiikka

S60-alusta on suunniteltu siten, että puhelimia on mahdollista käyttää yhdellä kädellä. Puhelimissa on perinteisesti ollut fyysinen näppäimistö ja näytöllä näkyvä käyttöliittymä on perustunut näytön alalaidassa olevien toimintonäppäinten (engl. Softkey) käyttöön. Sovelluksen tekijä voi määritellä, mitä toimintonäppäimet tekevät, mutta hyvin yleisesti vasemman puoleinen toimintonäppäin avaa valikon ja oikeanpuoleisella poistutaan sovelluksesta tai siirrytään edelliseen näkymään. Kun sovelluksesta poistutaan oikeaa toimintonäppäintä tai valikkoa käyttäen, sovellus suljetaan, eikä sitä jätetä taustalle. Jos sovellus halutaan jättää taustalle, jolloin sovelluksen käyttöä voi jatkaa myöhemmin samasta tilasta, pitää sovelluksesta poistua erityistä valikkonäppäintä käyttäen.

S60-alustan 5.0-versio vaikuttaa perusidealtaan hyvin samanlaiselta aiempiin versioihin verrattuna. Käyttöliittymän tyyli hyödyntää edelleen toimintonäppäimiä ja sovellusten käyttö onnistuu edelleen yhdellä kädellä, sormia käyttäen, vaikka puhelimessa olisikin kosketusnäyttö. Alla olevissa kuvissa 4 ja 5, on kuvakaappaukset S60:n 5.0-emulaattorin yhteystietosovelluksesta.



Kuva 4 *S60:n 5.0-emulaattorin yhteystietonäkymä, pystysuuntainen näyttö*



Kuva 5 *S60:n 5.0-emulaattorin yhteystietonäkymä, vaakasuuntainen näyttö*

Yllä olevissa kuvissa on toimintonäppäimien lisäksi nähtävissä muita näppäimiä näytön oikeassa ja alareunassa. Ne kuuluvat uuteen Toolbar APIin, jonka avulla sovellukseen saadaan lisättyä helposti ja nopeasti kosketettavissa olevia näppäimiä, joihin voidaan sijoittaa tärkeitä toimintoja. Uudet näppäimet parantavat käyttöliittymää huomattavasti, kun toimintoja ei tarvitse sijoittaa esimerkiksi valikkoon, josta niiden valitseminen olisi nykyistä selvästi vaivalloisempaa.

3.2 Kosketusnäyttö S60-alustalla

Tässä kappaleessa tutustutaan hieman tarkemmin kosketusnäyttöön S60-alustalla ja kerrotaan alustan 5.0-version mukana tulleista uusista ohjelmointirajapinnoista, jotka mahdollistavat paremman käyttöliittymän luomisen.

3.2.1 Yleistä kosketusnäytöstä S60-alustalla

Kosketusnäytön tuominen S60-alustalle on tuonut mukanaan monia uudistuksia. Eräs helpoiten havaittavista uudistuksista on tuki uudelle resoluutiolle, jonka myötä myös laitteiden näytön koko kasvaa. Alustan 5.0-versio tukee vanhan 240x320-resoluution lisäksi 640x360-resoluution omaavia kosketusnäyttöjä. /6/

Kosketusnäyttöä voi ohjata joko sormella tai kynällä ja se antaa käyttäjälle palautetta tactile feedback ominaisuuden avulla. Tactile feedback tarkoittaa sitä, että näytöllä näkyvät UI-komponentit voivat rekisteröidä tietyt alueet antamaan käyttäjälle palautetta uuden API:n avulla. Kun käyttäjä koskettaa esimerkiksi nappia, joka on rekisteröinyt itsensä antamaan palautetta, annetaan puhelimen värinätoiminnon avulla käyttäjälle tieto siitä, että painallus on tullut perille. /6/

Kosketusnäyttöä ajatellen, on S60:n 5.0-version käyttöliittymään tehty parannuksia ja lisäyksiä. Yksi lisäys on esimerkiksi uusi toolbar-komponentti, joka toimii työkalupalkkina, jonka avulla voidaan suorittaa tärkeitä toimintoja suoraan näytöllä näkyvää kohdetta koskettamalla. Myös tekstinsyöttö on uudistunut. Mahdollisuuksia on nyt perinteisten fyysisten numero- ja qwerty-näppäimistöjen lisäksi pieni qwerty-virtuaalinäppäimistö, käsialan tunnistukseen pohjautuva kirjoitus sekä koko näytön qwerty-näppäimistö. /6/

3.2.2 S60:n 5.0-versiossa esitellyt käyttöliittymärajapinnat

S60:n 5.0-versiossa julkaistiin useita uusia ohjelmointirajapintoja. Osa uusista rajapinnoista tuo alustaan uusia käyttöliittymäkomponentteja, jotka on suunniteltu kosketusnäyttöä ajatellen. Uudet käyttöliittymärajapinnat on esitetty taulukossa 2.

Ohjelmointirajapinta	Kuvaus
Choise List API	Mahdollistaa valintojen tekemisen pystysuuntaisesta listasta.
Generic Button API	Mahdollistaa sovellusten piirtää mielivaltaisen napin, jossa on ikoni tai teksti. Napin ominaisuudet voidaan määritellä, kuten onko se näkyvissä tai painettuna.
Hierarchical Lists API	Mahdollistaa datan esittämisen hierarkkisessa listassa käyttäen tekstiä ja grafiikkaa. Rajapinta tarjoaa kolme listatyyppiä: -yleiskäyttöinen hierarkkinen treelist -yhden sarakkeen lista, jossa alkio voidaan laajentaa ilman sisennystä -single-style tree, jolla kuvataan kansio- ja tiedostorakennetta. Tämä API julkaistiin jo S60 3rd Edition, Feature Pack 2 SDK:ssa, mutta toteutusta ei ole ollut missään laitteessa ennen S60:n 5.0-versiota.
Stylus Pop-up Menu API	Mahdollistaa sovellusten piirtää pop-up valikon, jossa on lista valintoja. Lista luodaan kohtaan, jossa kosketus näyttöön havaitaan.
Tactile Feedback Client API	Mahdollistaa sovellusten ja käyttöliittymä komponenttien rekisteröidä tactile feedback alueensa ja vastaanottaa tietoja kosketustapahtumista.
Title Pane Touch Observer API	Mahdollistaa sovellusten seurata käyttäjän painalluksia näytön Title Pane-alueelle.
Toolbar API	Mahdollistaa sovellusten rakentaa kiinteän tai liikkuvan työkalupalkin sovelluksen näkymään. Työkalupalkki koostuu AVKON-napeista, joita se voi sisältää korkeintaan neljä kappaletta.
Touch UI Utilities API	Tällä hetkellä tarjoaa vain palvelun, jolla tunnistetaan pitkä painallus kosketusnäytöllä.

Taulukko 2 *S60:n 5.0-version uudet käyttöliittymän ohjelmointirajapinnat* /2/

4 UIQ-ALUSTA

UIQ on suunniteltu siten, että laitevalmistajat voivat tehdä hyvin erilaisia puhelimia ja sovellukset toimivat niissä kaikissa. Kun sama koodi toimii kaikissa puhelimissa, on laitevalmistajien helppo tehdä erilaisia puhelimia. Myös kolmansien osapuolten työ helpottuu huomattavasti, kun sama koodi toimii kaikissa puhelimissa. Laitteet voivat erota toisistaan esimerkiksi näytön sekä näppäimistön osalta. Kuten kappaleessa 2.3 mainittiin, on UIQ tukenut kosketusnäyttöjä ensimmäisestä versiostaan alkaen. Seuraavissa kappaleissa perehdytään alustan 3.0-versioon. /8/

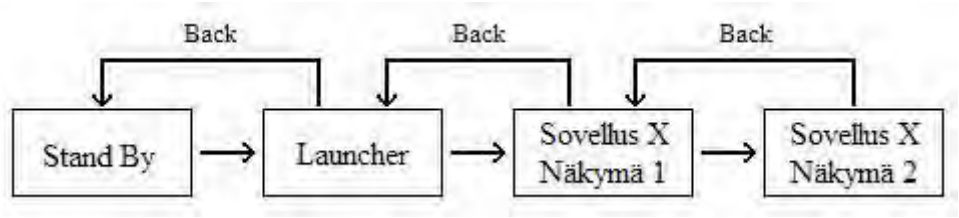
4.1 UIQ-sovellusten tyyli ja käyttölogiikka

Tässä kappaleessa tutustutaan UIQ-sovellusten yleiseen tyyliin sekä niiden toimintaan. Lisäksi kuvataan lyhyesti sovellusten rakennetta.

4.1.1 Yleistä

UIQ-sovellukset on tarkoitettu jäämään taustalle, kun niitä ei käytetä. Tällöin käyttäjä voi jatkaa sovelluksen käyttöä samasta tilanteesta, johon hän sen jätti poistuessaan siitä. Käyttäjälle ei anneta mahdollisuutta sulkea sovellusta, paitsi erillistä sovellusten hallintatyökalua käyttäen. Käytännössä tämä tarkoittaa sitä, että sovelluksissa ei ole valikossa eikä toimintonäppäimissä sovelluksen sulkemisvaihtoehtoa. /3/

Sovelluksesta poistuminen sekä näkymien välillä navigointi UIQ-alustalla tapahtuu erityisellä back-näppäimellä. Aina kun uusi näkymäluokka luodaan, kutsutaan sen kantaluokan CQikViewBasen rakentajaa antamalla sille parametrina toisen näkymän id, jonka halutaan toimivan uuden näkymän vanhempana. Kun jokaiselle näkymälle määritellään vanhempana toimiva näkymä, voidaan näkymien välillä siirtyä back-näppäintä painettaessa. Esimerkki sovelluksen navigointijärjestyksestä on nähtävissä kuvassa 6. /12/



Kuva 6 *Näkymien välillä navigointi UIQ-alustalla*

Kuvassa 6 on kuvattu siirtyminen puhelimen stand by-näkymästä launcherin kautta sovellukseen Sovellus X. Kun sovelluksen Näkymä 1 luodaan, annetaan kantaluokan rakentajalle parametrina arvo KNullViewId, joka tarkoittaa, että näkymällä ei ole samassa sovelluksessa vanhempaa, vaan näkymästä siirrytään sovelluksen ulkopuolelle back-näppäintä painettaessa. Kun luodaan toinen näkymä, annetaan parametrina edellistä näkymää vastaava id, jolloin se näkymä rekisteröityy toisen näkymän vanhemmaksi. Normaalisti back-näppäin välitetään kantaluokalle käsiteltäväksi. Sovelluskehys käsittelee komennon, tarkastamalla, mikä näkymä on asetettu nykyisen näkymän vanhemmaksi ja vaihtamalla näkymän siihen. /12/

Koska kaikki sovellukset jäävät oletuksena taustalle, kuluu muistia enemmän, kuin jos sovelluksia suljettaisiin. Jos puhelimen muisti käy vähiin, voi UIQ-alustan muistia hallitseva Memory Manager kehys pyytää sovellusta sulkeutumaan. Oletuksena kehys kutsuu document-luokan SaveL-funktiota, jossa sovelluksen tulee tallentaa kaikki tallentamattomat tietonsa. /3/

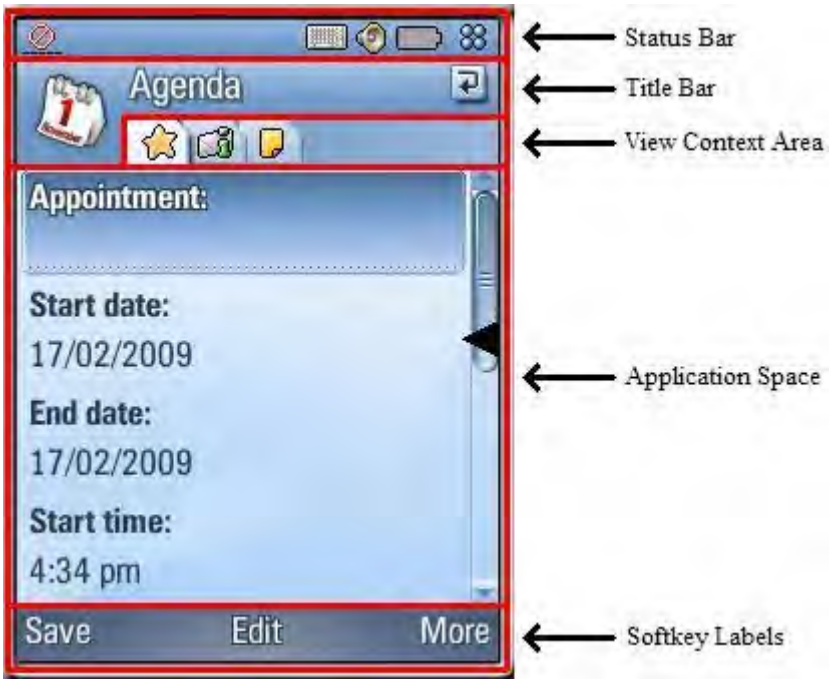
4.1.2 UIQ:n eri UI-konfiguraatiot

UIQ:n tuki erilaisille laitteille muodostuu UI-konfiguraatioista. Jokainen konfiguraatio muodostuu joukosta parametreja. Niin sanottuja käyttöliittymäparametreja ovat näytön resoluutio ja orientaatio, näytön kosketustuki sekä käyttöliittymän tyyli. UIQ:n 3.0-versio tukee kahta eri näytön resoluutiota: 240x320, sekä 240x256 pikseliä. Molemman kokoiset näytöt voivat olla käännettynä myös vaakatasoon, molempiin suuntiin. /8/

Käyttöliittymän tyyli voi UIQ-alustalla olla joko kynä- tai toimintonäppäintyyli. Kynätyyli on optimoitu kynän käyttöä varten, mutta sitä voidaan käyttää myös ilman kynää. Toimintonäppäintyyli on perinteinen, myös S60-alustan käyttämä tyyli, joka on optimoitu yhden käden käyttöä varten. Toimintonäppäintyyliä voidaan käyttää kosketusnäytöllä tai ilman, jolloin käyttöliittymä on hieman erilainen. Käytännössä tämä tyyli on hyvin lähellä S60-alustan tyyliä, mutta joitakin erojakin on. Seuraavissa kappaleissa perehdytään näihin kahteen tyyliin. /8/

4.1.2.1 Toimintonäppäintyylinen käyttöliittymä

Toimintonäppäintyylinen näkymä näyttöalueineen on nähtävissä kuvassa 7.



Kuva 7 *Toimintonäppäintyyli, tuki kosketusnäytölle*

Kuvassa 7 on ylimpänä Status Bar, joka sisältää tietoja järjestelmän tilasta, kuten akun varauksesta ja verkkoyhteyden tilasta. Tämän alueen sisältö määräytyy laitevalmistajan määrityksistä, eikä sovelluksen kehittäjä voi vaikuttaa siihen. Yllä olevassa kuvassa on kosketusnäyttöä tukeva tyyli, jossa Status Barissa näkyviä ikoneja voidaan koskettaa ja saada näin tarkempaa tietoa esimerkiksi akun varauksesta. /10/

Status Barin alapuolella on Title Bar eli otsikkopalkki. Tämä alue sisältää yleensä ainakin sovelluksen nimen tai muun tekstin sekä sovelluksen ikonin. Kuvassa näkyy, että kosketusnäyttöä hyödyntävässä toimintonäppäintyylissä Title Barissa on paluunappi, jota käytetään kappaleessa 4.1.1. kuvatun mukaiseen navigointiin näkymien välillä. /10/

View Context Area on alue, jolla sovellus voi näyttää monenlaista tietoa tai esimerkiksi helpottaa navigointia näkymien välillä välilehtien avulla. Alueelle voidaan lisätä myös esimerkiksi ikoneita tai jonkin toiminnon etenemistä kuvaava palkki. /10/

Application Space on alue, jolle sovelluksen näkymät piirretään. Tämä alue ei poikkea juurikaan S60-alustan versiosta. Ainoa ero tässä alueessa eri konfiguraatioiden välillä on vierityspalkin leveys, joka on kosketusnäyttöä käytettäessä hieman suurempi. Kuten S60-alustalla, voidaan UIQ-alustallakin ottaa koko näyttö käyttöön sovellusta varten, mutta silloin pitää olla hyvin tarkkana käyttöliittymän käyttäjäystävällisyyden kanssa. /10/

Alimmaisena toimintonäppäintyylisessä konfiguraatiossa näytöllä on Softkey Labels alue, joka sisältää käytännössä vain otsikot, jotka kuvaavat, mitä tapahtuu, kun nappia painetaan. Toimintonäppäimet toimivat UIQ-alustalla samaan tapaan kuin S60-alustallakin, mutta näppäinten järjestys on erilainen. UIQ-alustalla valikko avataan oikeanpuoleista näppäintä painamalla. /10/

Kuten aiemmin mainittiin, voidaan toimintonäppäintyyliä käyttää myös ilman kosketusnäyttöä. Kuvassa 8 on esitetty näkymä toimintonäppäintyylistä ilman kosketusnäyttöä.

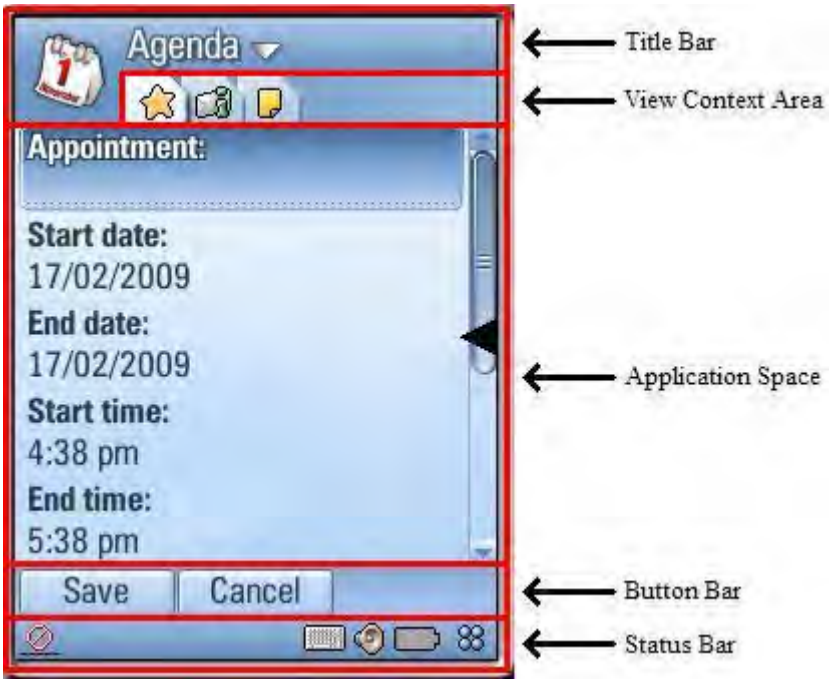


Kuva 8 *Toimintonäppäintyyli, ei tukea kosketusnäytölle*

Kuten kuvasta 8 voidaan huomata, ovat näkymät hyvin samankaltaisia kosketusnäytöllä ja ilman. Voidaan sanoa, että kosketusnäyttö tuo tiettyjä lisämahdollisuuksia käyttöliittymään, jotka voivat helpottaa toimintojen tekemistä ja parantaa käyttäjäystävällisyyttä. Ilman kosketusnäyttöä, ei Status Barin kautta saada suoraan tarkempia tietoja järjestelmän tilasta. Myös paluunäppäin puuttuu ja on toteutettu tällaisessa laitteessa fyysisenä näppäimenä.

4.1.2.2 Kynätyylinen käyttöliittymä

Kynätyylin erot toimintonäppäintyyliin verrattuna on melko helppo havaita katsomalla kuvaa 9. Näytön alueet ovat pääosin samat, mutta niiden järjestys on osittain vaihtunut. Merkittävin ero tyylien välillä on se, että kynätyylissä toimintonäppäimet on korvattu näppäinpalkilla (Button Bar). Kyseisellä alueella on yleensä toimintonäppäintyylistäkin tuttu paluunappi sekä muita nappeja, joita sovelluksen tekijä sinne on halunnut sijoittaa. /10/



Kuva 9 *Kynätyyli, pystysuuntainen näyttö*

Title Barissa on kynätyylissä heti sovelluksen nimen jälkeen nuoli-ikoni, jota koskettamalla saadaan näkyviin valikko. Valikko on sama kuin toimintonäppäintyylissä oikeanpuoleista toimintonäppäintä painettaessa aukeava valikko. Jotta kosketusnäytön käyttö olisi helppoa ja nopeaa, on UIQ:ssa kuitenkin mahdollista määritellä valikon alkioille tietyllä lipulla, halutaanko niiden olevan mieluummin valikossa vai Button Barissa. Jos alkiolle on lipulla määritelty, että sen halutaan mieluummin olevan Button Barissa helposti ja nopeasti valittavissa, niin silloin Button Bariin tulee tätä komentoa vastaava nappi, eikä komentoa löydy Title Barista aukeavasta valikosta. /10/

Kuten aiemmin mainittiin, tukee UIQ myös näytön käyttöä vaakasuunnassa. Kuvassa 10 on esimerkki UIQ:n näkymästä, kun näyttöä käytetään vaakasuunnassa.



Kuva 10 *Kynätyyli, vaakasuuntainen näyttö*

Yllä olevassa kuvassa näkyy myös hyvin, että Button Barissa voidaan käyttää tekstin sijaan ikoneita näppäimissä, jolloin kyseiselle alueelle voidaan sijoittaa enemmän toimintoja. /10/

4.1.2.3 Eri UI-konfiguraatioihin varautuminen

UIQ-alusta tukee eri käyttöliittymäkonfiguraatioita ja näin ollen erilaisia puhelimia. Jotkut puhelimet tukevat useita eri UI-konfiguraatioita. Konfiguraatioiden vaihtaminen on joissain tapauksissa mahdollista tehdä sovelluksen sisältä mutta joissakin tapauksissa vaihto voi tapahtua vain jonkun järjestelmän tapahtuman yhteydessä. Sovellus voi esimerkiksi vaihtaa näytön orientaatiota mutta esimerkiksi joissakin puhelimissa oleva aukeava mekanismi, joka paljastaa fyysisen näppäimistön, saa aikaan konfiguraation vaihdon jota ei voi sovelluksesta ohjelmallisesti tehdä. /11/

Eri käyttöliittymäkonfiguraatioita varten voidaan sovelluksen resursseissa määritellä erilliset näkymät ja komennot, joita käytetään eri konfiguraatioilla. Normaalisti sovellusten ei tarvitse määritellä eri konfiguraatioille eri näkymiä ja komentoja. Joissakin erityistapauksissa voidaan kuitenkin haluta sovelluksen näyttävän erilaiselta tai sisältävän eri komentoja eri konfiguraatioilla. Jokainen sovelluksen näkymä voi määritellä, mitä käyttöliittymäkonfiguraatioita se tukee.

Määrittely tehdään resursseissa QIK_VIEW_CONFIGURATIONS-rakenteen avulla. Kuvassa 11 on esitetty, kuinka tämä määrittely tehdään. /11/

```
RESOURCE QIK_VIEW_CONFIGURATIONS r_ui_configurations
    {
    configurations =
        {
        QIK_VIEW_CONFIGURATION
            {
            ui_config_mode = KQikPenStyleTouchPortrait;
            command_list = r_Hello_commands;
            view = r_Hello_layout;
            },
        QIK_VIEW_CONFIGURATION
            {
            ui_config_mode = KQikPenStyleTouchLandscape;
            command_list = r_Hello_commands;
            view = r_Hello_layout;
            },
        QIK_VIEW_CONFIGURATION
            {
            ui_config_mode = KQikSoftkeyStyleTouchPortrait;
            command_list = r_Hello_commands;
            view = r_Hello_layout;
            },
        QIK_VIEW_CONFIGURATION
            {
            ui_config_mode = KQikSoftkeyStylePortrait;
            command_list = r_Hello_commands;
            view = r_Hello_layout;
            },
        QIK_VIEW_CONFIGURATION
            {
            ui_config_mode = KQikSoftkeyStyleSmallPortrait;
            command_list = r_Hello_commands;
            view = r_Hello_layout;
            }
        };
    }
```

Kuva 11 *Näkymän tukemien UI-konfiguraatioiden määrittely*

Yllä olevassa kuvassa on tavallisen graafisen UIQ-sovelluksen tukemien käyttöliittymäkonfiguraatioiden esittelyt. Eri konfiguraatiot määritellään QIK_VIEW_CONFIGURATION-määrittelyn avulla. Jokaiselle konfiguraatiolle voidaan tässä määritellä, mitkä komennot ja mikä näkymä on käytössä milläkin konfiguraatiolla. Yllä olevassa esimerkissä käytetään kaikilla konfiguraatioilla samoja komentoja ja samaa näkymää. /11/

Jos puhelin käyttää käyttöliittymäkonfiguraatiota, jolle ei ole sovelluksen resursseissa määrittelyä, etsii järjestelmä sopivimman määritellyn konfiguraation ja käyttää sitä sovelluksen alustamiseen. Sovelluksen on mahdollista pyytää järjestelmää vaihtamaan haluttuun konfiguraatioon mutta tämä ei välttämättä ole mahdollista, jos puhelin ei tue kyseistä konfiguraatiota. Järjestelmä määrää siis aina, mitä konfiguraatiota käytetään. Koska järjestelmä etsii aina sopivimman konfiguraation, ei sovelluksessa tarvitse huolehtia kaikista konfiguraatioista, vaan sama koodi toimii käytännössä kaikissa puhelimissa. /11/

5 WIRELESS TEST FRAMEWORK

Wireless Test Framework on Sasken Finland Oy:ssä Symbian-alustalle diplomi- ja opinnäytetöinä kehitetty testaustyökalu. Sen avulla voidaan tehdä yksikkö-moduuli- ja integrointitestausta. Sillä voidaan siis testata esimerkiksi jonkun luokan toimintaa ja varmistaa, että toiminta on haluttua.

Testikehykseen on aiemmin tehty käyttöliittymä S60-alustalle. Sovelluksen toiminta haluttiin laajentaa toimimaan myös UIQ-alustalla, joka oli tämän tutkintotyön tarkoitus. Koska S60- ja UIQ-alustat pohjautuvat Symbian-käyttöjärjestelmään, oli suurin työ siirtää käyttöliittymä UIQ:lle, johtuen UI:ssa käytettävistä alustojen omista luokista. Seuraavissa kappeleissa tutustutaan hieman tarkemmin testikehykseen ja erityisesti sen käyttöliittymään.

5.1 Yleistä testikehyksestä

Testikehyksen tärkeimmän toiminnallisuuden toteuttaa TestServer-komponentti, jota käytetään asiakas-palvelin-periaatteella. TestServer toteuttaa esimerkiksi testimoduulien ja –tapausten latauksen, testien ajamisen ja tulosten raportoinnin. Testitapaukset sijaitsevat testimoduuleissa, jotka ovat dll-tyyppisiä kirjastoja. Yksi kirjasto voi sisältää esimerkiksi jonkun tietyn luokan testaamiseen kirjoitettuja testitapauksia.

TestServerin toimintaa ohjataan käyttöliittymän avulla. Käyttäjä voi valita haluamansa testitapaukset haluamistaan testikirjastoista ja muodostaa näin testisetin. Kun testejä ajetaan, ilmoittaa TestServer testien etenemisestä tarkkailijaluokalleen. Kun tarkkailijaluokka saa tiedon esimerkiksi testitapauksen ajon valmistumisesta, päivittää se näkymän vastaamaan tätä tilannetta.

Testikehykseen on tehty TestServerin ja UI:n lisäksi muitakin osia, jotka ovat ScriptingModule ja SynchronizationModule. Näiden moduulien avulla testitapauksia voidaan skriptata ja synkronoida, mutta testien ajaminen onnistuu ilmankin näiden moduulien käyttöä. Tässä tutkintotyössä ei käsitellä muita testikehyksen osia kuin käyttöliittymää ja sen kommunikointia TestServerin kanssa.

5.2 Käyttöliittymä

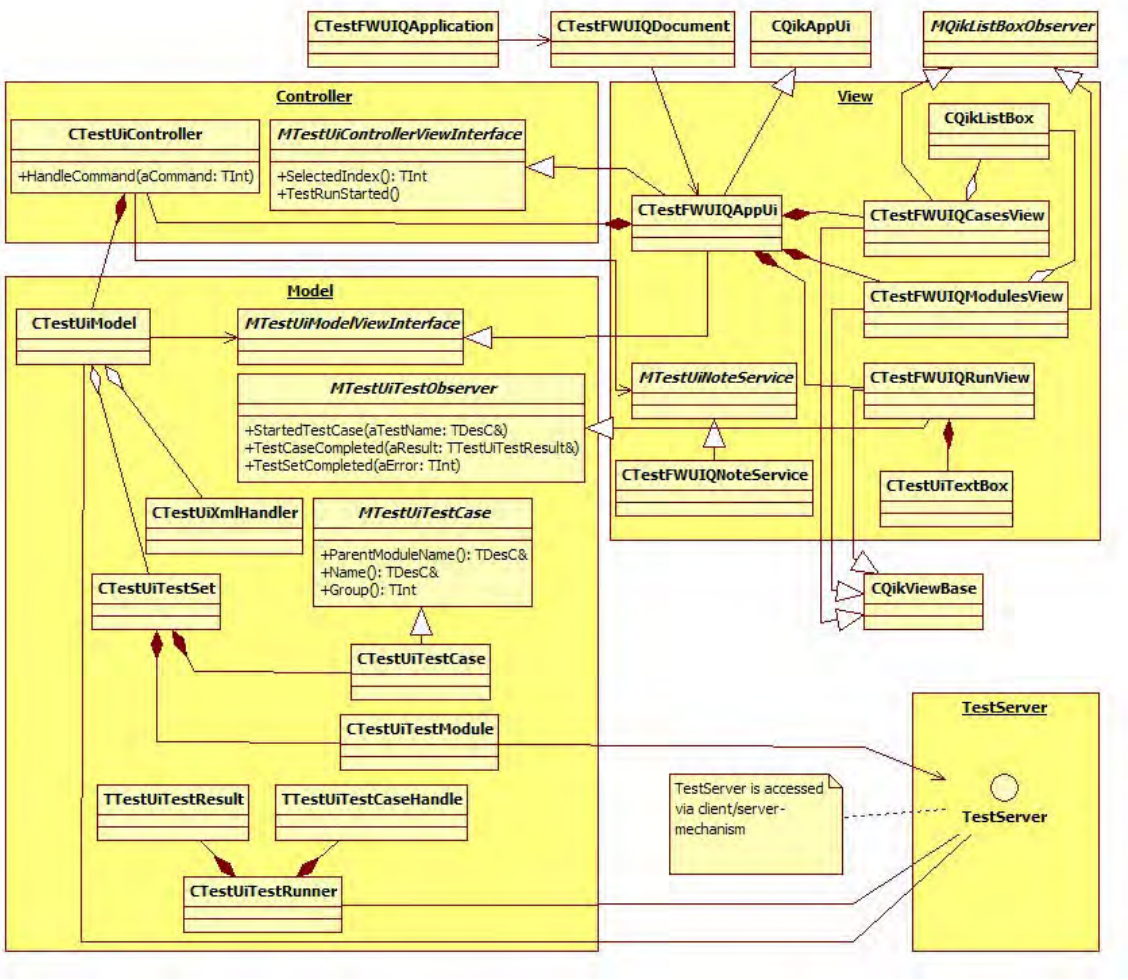
Testikehyksen käyttöliittymän UIQ-versio perustuu sovelluksen S60-versioon. Täten käyttöliittymä on saman tyyppinen ja sisältää samat ominaisuudet molemmilla alustoilla. Seuraavissa kappaleissa perehdytään tarkemmin käyttöliittymän rakenteeseen ja toimintaan.

5.2.1 Rakenne

Käyttöliittymä noudattaa MVC-arkkitehtuuria ja sisältää useita luokkia, jotka muodostavat malli-, näkymä- ja kontrollerimoduulit. Kontrolleri käsittelee kaikki näkymiltä tulevat komennot ja ohjaa mallin toimintaa niiden perusteella. Kontrolleri pitää huolen myös näkymien päivittämisestä tarpeen mukaan.

Malli sisältää tiedon sovelluksen tilasta ja kommunikoi käyttöliittymän ulkopuolelle testitapausten ja -moduulien käsittelyn yhteydessä. Kontrolleri voi mallin avulla esimerkiksi lisätä testimoduuleja ja –tapauksia, ajaa testitapauksia ja poistaa moduuleja ja tapauksia. Näkymätkin ovat yhteydessä malliin vakioviitteen kautta. Näkymät voivat kysyä mallilta esimerkiksi testisetin sisällön, jonka avulla ne päivittävät itsensä vastaamaan sovelluksen tilaa.

Näkymiä käyttöliittymässä on kolme, joista kerrotaan tarkemmin seuraavissa kappaleissa. Kuvassa 12 on nähtävissä käyttöliittymän luokkakaavio.



Kuva 12 *Käyttöliittymän luokkakaavio*

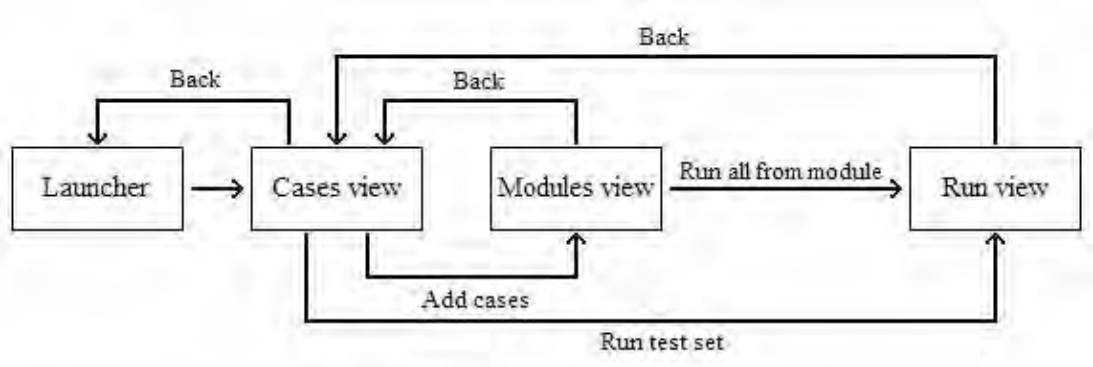
Sovelluksen AppUi-luokka suorittaa lähinnä vain näkymien hallintaa ja välittää käyttäjän komennot kontrollerille käsiteltäväksi. AppUi toteuttaa MTestUiControllerViewInterface- ja MTestUiModelViewInterface-rajapinnat, joiden avulla kontrolleri ja malli ilmoittavat näkymämoduulille tapahtumista. Esimerkiksi Update-funktiossa AppUi välittää päivityskäskyn aktiivisena olevalle näkymälle, jolloin käyttäjälle näkyvä näkymä saadaan päivitettyä.

5.2.2 Näkymät sekä toiminnallisuudet

Testikehyksen käyttöliittymä koostuu siis kolmesta näkymästä. Näkymät ovat tapausnäkymä, moduulinäkymä sekä ajonäkymä. Koska käyttöliittymä suunniteltiin kosketusnäyttöä hyödyntäville laitteille, on näkymien suunnittelussa otettu huomioon käytön helppous sekä kosketusnäytön tuomat uudet mahdollisuudet.

Käyttöliittymän suunnittelussa lähdettiin siitä, että toimintojen valinnat pitäisi pystyä tekemään helposti ja valikoiden käyttöä pitäisi saada rajoitettua. Eniten toimintoja on sovelluksen moduuli- ja tapausnäkymissä. Nämä näkymät ovat saman tyyppiset, sisältäen listat, jotka kuvaavat ladattuja moduuleja ja testisettiä. UIQ:n monipuolinen listaluokka CQikListBox tarjosi mahdollisuuden sisällyttää toimintoja suoraan listan alkioiden yhteyteen, sikäli kun toiminto kohdistui johonkin tiettyyn moduuliin tai testitapaukseen. Tämän avulla sovelluksen käyttöä saatiin helpommaksi, kun kaikkia valintoja ei tarvitse tehtä valikon kautta. Komentojen käytöstä listan kautta kerrotaan lisää myöhemmin.

Koska sovelluksessa on kolme näkymää, piti suunnitella järkevin tapa navigoida näkymien välillä back-näppäintä käyttäen. Koska tapausnäkymän kautta voidaan ladata testisettejä sekä aloittaa setin ajaminen, voidaan sitä pitää sovelluksen päänäkymänä. Kuvassa 13 on esitetty testikehyksen näkymien välillä navigointi.



Kuva 13 *Testikehyksen näkymien välillä navigointi*

Sovelluksen käynnistyessä tapausnäkymä on aktiivinen. Sovelluksen muista näkymistä palataan back-näppäimellä aina kyseiseen näkymään. Kuvassa 13 on nähtävissä myös komentoja, joiden seurauksena näkymiä vaihdetaan. Näistä

komennoista ja näkymien toiminnallisuuksista sekä kommunikoinnista muiden luokkien kanssa kerrotaan lisää seuraavissa kappeleissa.

5.2.2.1 Tapausnäkymä

Tapausnäkymä sisältää listan tämän hetkisen testisetin testitapauksista. Näkymä antaa mahdollisuuden ajaa testisetin, poistaa testitapauksia, ladata tallennetun testisetin tai siirtyä moduulinäkymään. Kun tapausnäkymään siirrytään, kysyy näkymä mallilta nykyiseen testisettiin lisätyt testitapaukset ja päivittää listan vastaamaan niitä. Jokainen listan alkio vastaa yhtä testitapausta. Kuvassa 14 on nähtävissä tapausnäkymä tilanteessa, jossa testisetti sisältää viisi testitapausta.



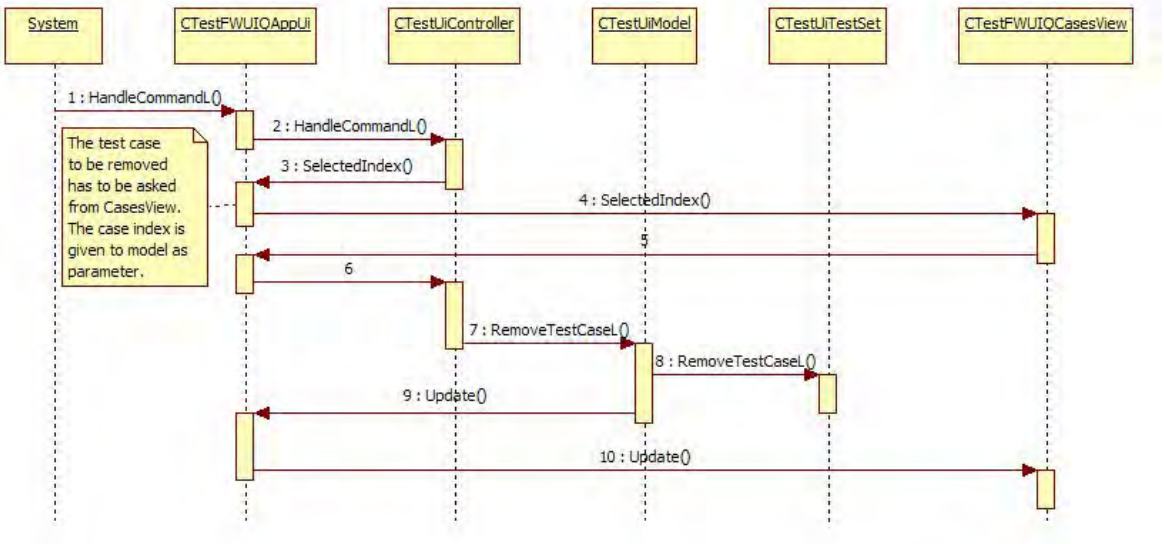
Kuva 14 *Tapausnäkymä, kosketusnäyttöä hyödyntävä toimintonäppäintyyli*

Listan alkiot sisältävät kaksi riviä, joista ylempi kertoo testitapauksen nimen ja toinen toiminnon, joka suoritetaan, kun listan alkio valitaan. Kun listan alkio valitaan, poistetaan kyseinen testitapaus testisetistä. Kuten kuvasta 14 näkyy, on listan alkioita mahdollista valita useita kerralla mutta toteutusta usean tapauksen valikoivaan poistoon ei ole vielä tehty. Vaikka testitapauksia ei voi poistaa valikoivasti, on valikossa kuitenkin mahdollisuus poistaa kaikki testitapaukset

kerralla. Listan alkion kosketus käsitellään näkymän HandleListBoxEventL-funktiossa, koska näkymä on rekisteröity kyseisen listan tarkkailijaksi.

HandleListBoxEventL-funktiossa luodaan ja laukaistaan testitapauksen poistoa vastaava näppäintapahtuma eli periaatteessa toimitaan samoin kuin FEP:t, jotka tekevät esimerkiksi kosketustapahtumalle esikäsittelyä ja luovat sitten sopivan näppäintapahtuman. Näppäintapahtuma käsitellään AppUi:n HandleCommandL-funktiossa.

UIQ:ssa näppäintapahtumaa kuvastaa CQikCommand-tyyppinen olio, joka poikkeaa S60-alustan käyttämästä TInt-tyypistä. UIQ:n komentotyyppi sisältää enemmän tietoja komennosta, kuten tyyppitiedon, joka määrittää, missä komento näytetään milläkin käyttöliittymäkonfiguraatiolla. Koska eri alustat käyttävät erityyppisiä komentoja, on UIQ-version AppUi:n muunnettava komento TInt-tyyppiseksi, jolloin kontrolleriin ei vaadita muutoksia komentojen käsittelyn suhteen, vaan sama toteutus toimii eri alustoilla. Kuvassa 15 on nähtävissä yksinkertaistettu testitapauksen poistosekvenssi.



Kuva 15 *Testitapauksen poistaminen*

Testitapauksen poisto etenee seuraavasti: AppUi saa näkymässä luodun näppäinkomennon käsiteltäväkseen ja välittää sen kontrollerille. Kontrolleri kysyy AppUi:lta valitun testitapauksen indeksin ja käskee tämän jälkeen mallia

poistamaan testitapauksen. Kuten kappaleessa 5.2.1 kerrottiin, AppUi toteuttaa sekä MTestUiModelViewInterface- että MTestUiControllerViewInterface-rajapinnat ja toimii täten rajapintana näkymien ja kontrollerin sekä mallin välillä. AppUi:lla on tiedossa aktiivisena oleva näkymä, jolle se välittää SelectedIndex-kutsun.

Malli välittää testitapauksen poistopyynnön CTestUiTestSet-luokalle, joka poistaa tapauksen. Tämän jälkeen näkymä pitää päivittää, koska testitapaus on poistettu. Näkymien päivitys tapahtuu samaan tyyliin, kuin indeksin kysely eli AppUi kutsuu aktiivisen näkymän Update-funktiota, jossa näkymä päivitetään kysymällä tarvittavat tiedot mallilta.

5.2.2.2 Moduulinäkymä

Moduulinäkymässä on lista testikehykseen ladatuista testimoduuleista. Näkymässä on mahdollista ladata lisää testimoduuleja, poistaa moduuleja, lisätä testisettiin testitapauksia, ajaa kaikki tapaukset halutusta moduulista ja siirtyä tapausnäkymään.

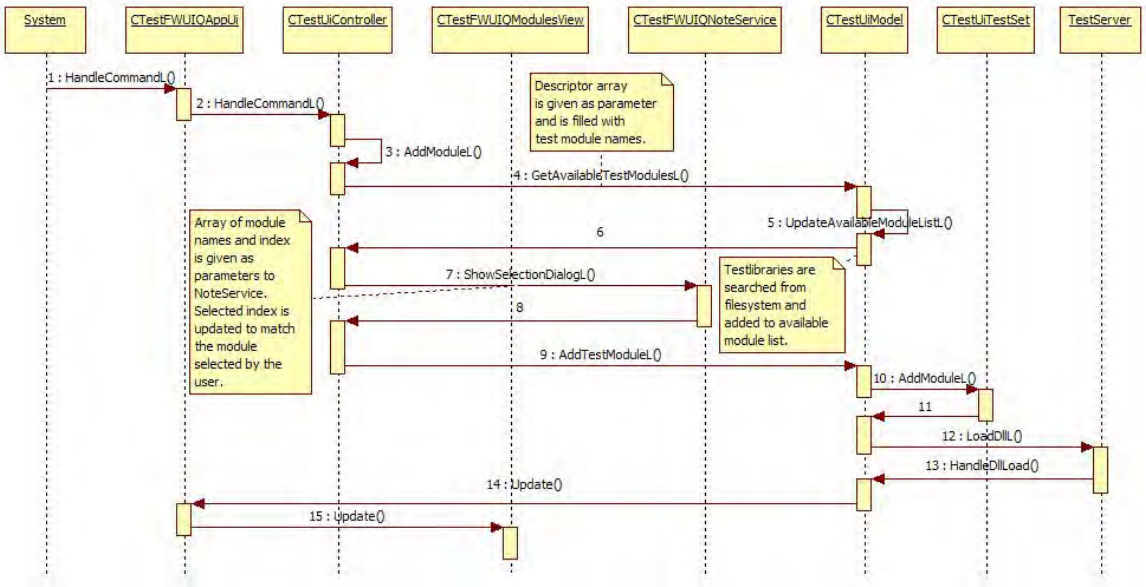
Kuten tapausnäkymässäkin, on toimintoja sisällytetty listan alkioiden yhteyteen, jotka vastaavat ladattuja moduuleja. Moduulin poisto, testitapauksen lisäys ja kaikkien tapausten ajo tietystä moduulista on mahdollista tehdä suoraan listan avulla, ilman valikosta tehtäviä valintoja. Moduulista on myös mahdollista lisätä kaikki testitapaukset testisettiin kerralla. Kuten tapausnäkymässäkin, myös moduulinäkymässä listan alkioiden toinen rivi määrää, mikä toiminto tehdään, kun alkiota kosketetaan. Haluttua toimintoa voidaan muuttaa alkion toisen rivin päädyissä olevia nuolia koskettamalla. Kuvassa 16 on nähtävissä moduulinäkymä, kun testikehykseen on ladattu kaksi testimoduulia.

FIFTH THIRD EX1062
PART 2 of 3
Page 342 of 379



Kuva 16 *Moduulinäkymä, kosketusnäyttöä hyödyntävä toimintonäppäintyyli*

Uuden testimoduulin lataaminen tapahtuu valitsemalla valikosta testimoduulin lataamistoiminto. Valikoista tehtäville komennoille ei tehdä näkymissä mitään käsittelyä vaan annetaan AppUi:n käsitellä ne ja välittää komento kontrollerille. Kuvassa 17 on nähtävissä testimoduulin lataamissekvenssi.



Kuva 17 *Testimoduulin lataaminen*

Testimoduulin lataaminen etenee seuraavasti: AppUi välittää komennon kontrollerille, joka kysyy saatavilla olevia testimoduuleja mallilta. Kontrolleri antaa mallille parametrina muuttujan, johon malli lisää saatavilla olevien testimoduulien nimet. Malli päivittää moduulilistan etsimällä dll-tyyppisiä testikirjastoja tietystä paikasta.

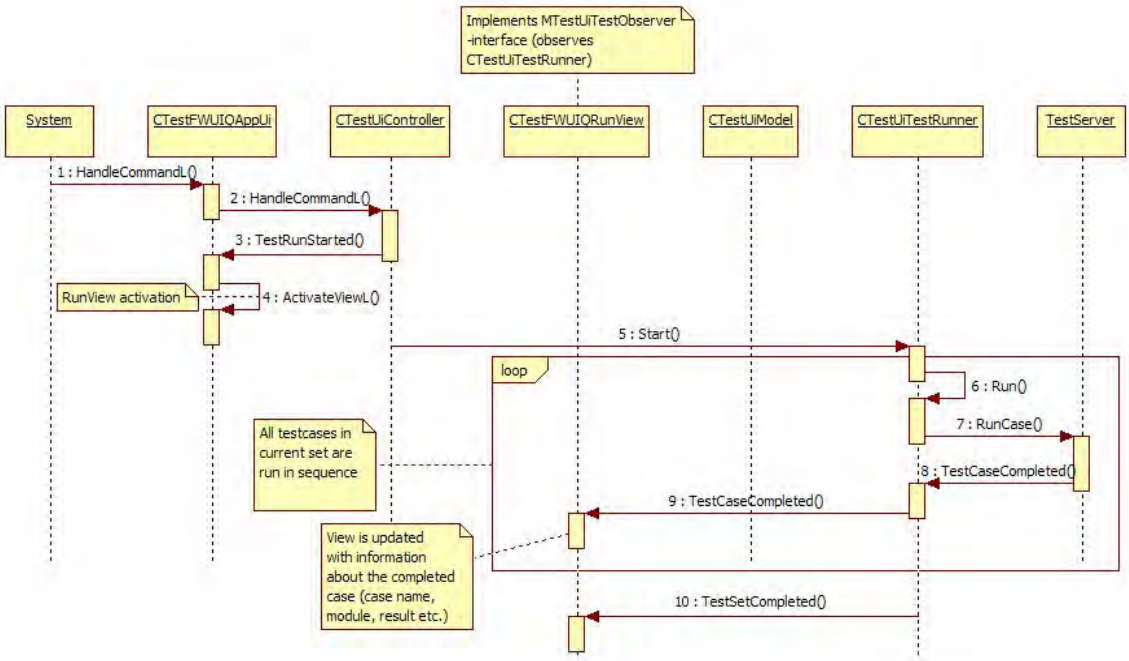
Kun kontrolleri on saanut tietoonsa saatavilla olevat moduulit, kutsuu se NoteServicen ShowSelectionDialogL-funktiota. NoteService näyttää käyttäjälle dialogin, joka sisältää listan moduuleista. Kun käyttäjä valitsee haluamansa moduulin, asetetaan kyseisen moduulin indeksi parametrina saatuun muuttujaan.

Tämän jälkeen kontrolleri käskee mallia lisäämään kyseisen moduulin antamalla valitun indeksin parametrina. Malli kutsuu TestSet-luokan AddModule-funktiota, jossa uusi moduuli lisätään testisettiin. Seuraavaksi pitää tehdä moduulin varsinainen lataus, joka tehdään TestServerin avulla. TestServer on aktiiviolio, joka latauksen valmistuttua kutsuu mallin HandleDllLoad-funktiota. Jos moduulin lataus onnistui, päivitetään näkymä. Jos lataus ei onnistunut, poistetaan moduuli testisetistä.

5.2.2.3 Ajonäkymä

Ajonäkymä sisältää testikehykseen itse tehdyn tekstikentän, johon ajettavien testien tulokset tulostetaan. Näkymässä ei tällä hetkellä ole muita toimintoja, kuin testisetin ajamisen uudelleen käynnistys, sekä siirtyminen takaisin tapausnäkymään. Ajonäkymään siirtyminen, sekä testien ajaminen on kuvattu kuvassa 18.

FIFTH THIRD EX1062
PART 2 of 3
Page 344 of 379



Kuva 18 *Testitapausten ajaminen*

Testisetin ajaminen etenee seuraavasti: Kontrolleri saa käsiteltäväkseen testisetin ajamiskomennon, jonka käyttäjä on laukaissut tapausnäkymästä tai ajonäkymästä. Sama sekvenssi toistuu myös silloin, kun käyttäjä valitsee ”Run all from module”-komennon moduulinäkymässä. AppUi saa tiedon testien ajamisen aloittamisesta kontrollerilta ja aktivoi ajonäkymän, johon testien ajon aikana tulostetaan tietoja.

Seuraavaksi kontrolleri kutsuu CTestUiTestRunner-tyyppisen olion Start-funktiota, jossa olio asetetaan aktiiviseksi ja käsketään TestServeriä ajamaan testitapaus. Kun testitapaus on ajettu, saa CTestUiTestRunner siitä tiedon ja kutsuu tarkkailijaluokkansa TestCaseCompleted-funktiota. Kyseisen olion tarkkailijana toimii ajonäkymän ilmentymä, joka päivittää itsensä aina tapauksen ajon valmistuttua. Käytännössä näkymä tulostaa tekstikenttäänsä ajetun testitapauksen nimen ja tiedon siitä, onnistuiko testi vai ei. Kaikkien testisetissä olevien testitapausten ajo tehdään edellä kuvatun mukaisesti peräkkäin.

Kun testisetin kaikki testitapaukset on ajettu, saa ajonäkymä tästä tiedon ja tulostaa näytölle testien ajamisen päättyneen. Lisäksi näytölle tulostetaan läpi menneiden ja

epäonnistuneiden testitapausten määrät. Kuvassa 19 on nähtävissä ajonäkymä, kun kaikki viisi testisetissä ollutta testitapausta on ajettu.



Kuva 19 *Ajonäkymä, kosketusnäyttöä hyödyntävä toimintonäppäintyyli*

5.2.3 Käyttöliittymän parannukset

Sen lisäksi, että tässä opinnäytetyössä tuli hyödyntää UIQ-alustan ja kosketusnäytön mukanaan tuomia mahdollisuuksia käyttöliittymän suunnittelussa, piti sovelluksen käyttöliittymää parantaa myös joidenkin käyttöä helpottavien lisäysten avulla. Käytännössä suurin osa lisäyksistä on hyvin pieniä mutta ne parantavat käyttäjäystävällisyyttä huomattavasti. Näitä pieniä parannuksia ovat:

- moduulista on nyt mahdollista lisätä kaikki testitapaukset testisettiin kerralla
- testisetistä voidaan poistaa kaikki testitapaukset kerralla
- testisetin ajaminen voidaan käynnistää uudestaan suoraan ajonäkymästä

Edellä mainittujen parannusten lisäksi toteutettiin käyttöliittymään testisettien lataus xml-muotoisesta tiedostosta. Kun testisettien lataus tehdään yksinkertaiseen xml-tiedostoon perustuen, on testaajan helppo luoda haluamiaan testisettejä

tietokoneella ja käyttää niitä sitten puhelimessa. Xml-tiedostoon voidaan tallentaa käytännössä rajaton määrä testisettejä, jotka voivat sisältää testitapauksia useista eri moduuleista. Testisetistä kuvataan xml-tiedostossa setin nimi, testisettiin kuuluvien testitapausten nimet sekä moduulit, joihin testitapaukset kuuluvat.

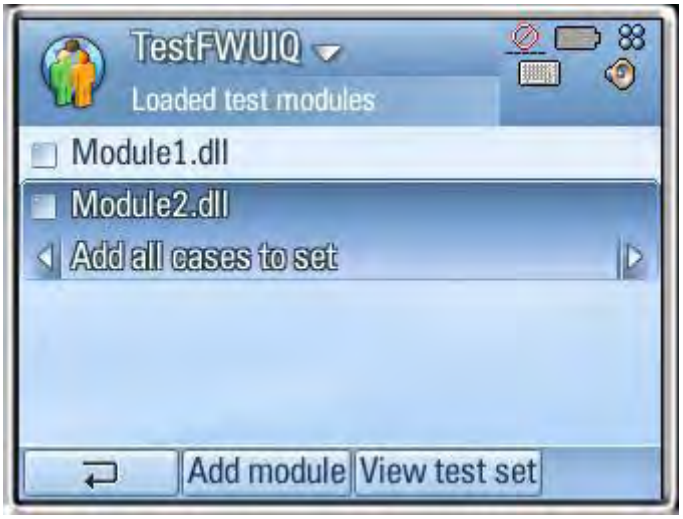
5.2.4 Toiminta eri UI-konfiguraatioilla

Eri käyttöliittymäkonfiguraatioita varten ei tässä sovelluksessa ole tehty mitään muokkauksia, vaan kaikilla konfiguraatioilla käytetään samoja komentoja ja näkymiä. Optimointiin eri konfiguraatioita varten ei nähty tarvetta, koska sovelluksen käyttöliittymä toimi kaikilla testatuilla konfiguraatioilla hyvin emulaattorilla testattaessa.

Moduuli- ja testitapausnäkymät koostuvat CQikListBox-tyyppisestä listasta, joka on hyvin muuntautuva ja suunniteltu toimimaan eri käyttöliittymäkonfiguraatioilla. Testitapausten ajonäkymässä on käytössä testikehykseen itse tehty tekstikontrolli. Myös tämä kontrolli mukautuu hyvin eri konfiguraatioihin. Alla olevissa kuvissa 20, 21 ja 22 on kuvakaappauksia sovelluksen ajamisesta kynätyylin ollessa käytössä. Kuvien tilanteissa näyttöä käytetään vaakasuunnassa. Kuvat vastaavat kappaleissa 5.2.2.1, 5.2.2.2 ja 5.2.2.3 esitettyjä tilanteita.



Kuva 20 *Tapausnäkymä, kynätyyli, vaakasuuntainen näyttö*



Kuva 21 *Moduulinäkymä, kynätyyli, vaakasuuntainen näyttö*



Kuva 22 *Ajonäkymä, kynätyyli, vaakasuuntainen näyttö*

Kuten kuvista nähdään, kynätyyliä käytettäessä valikon komennot on sijoitettu Button Bariin. Tämä on toimiva ratkaisu, koska toiminnot saa tehtyä helposti ilman valikoita.

5.3 Ongelmakohdat

Tässä kappaleessa perehdytään joihinkin työn tekemisessä esiintyneisiin ongelmiin ja esitetään ratkaisut niiden selvittämiseksi.

5.3.1 Käytöstä poistettu dialogi

Sovelluksen S60-versiossa käytetään CAknSelectionListDialog-tyyppistä oliota valintalistan toteutukseen. Kyseinen luokka periytyy CAknDialog-luokan kautta CEikDialog-luokasta, jota ei UIQ:n 3.0-versiossa enää tueta.

CEikDialogin korvaajiksi UIQ:n 3.0-versiossa tuli CQikSimpleDialog ja CQikViewDialog. CQikViewDialog on tavallisen näkymän ja dialogin yhdistelmä. Se mahdollistaa dialogien käytöstä tutun synkronisen koodin suorituksen sekä näkymien tuen monipuolisille konfiguraatioille. Synkroninen koodin suoritus tarkoittaa sitä, että koodin suoritus jää odottamaan dialogista paluuta. /4/

Sovelluksen valintalista toteutettiin periyttämällä luokka CQikViewDialogista sekä MQikListBoxObserverista. Dialogi sisältää CQikListBox-tyyppisen listan, johon halutut listan alkiot lisätään. Luokan rakentajalle annetaan parametrina taulukko listan alkioista, joilla luotava lista täytetään. Kun käyttäjä valitsee listalta jonkun alkion, palauttaa dialogi valitun alkion indeksin kutsujalle.

5.3.2 UIQ yhtiön konkurssi

UIQ yhtiö hakeutui konkurssiin tämän työn tekemisen aikana. Konkurssin seurauksena esimerkiksi developer.uiq.com-sivusto lopetti toimintansa. Sivusto oli saman tapainen kuin Forum Nokian sivut eli se sisälsi dokumentaatiota sekä keskustelualueita sovelluskehitykseen liittyen.

Koska UIQ-alustan käyttäjä- ja kehittäjäkunta on selvästi pienempi kuin S60-alustalla, oli konkurssin jälkeen hankala löytää apua Internetistä. Erityisesti sovelluskehittäjien keskustelupalstoja oli vaikea löytää. UIQ:n SDK:n dokumentaatio on kuitenkin hyvää ja kattavaa.

UIQ:n konkurssi ei tarkoita sitä, että kaikki UIQ-alustan hyvät ominaisuudet menisivät hukkaan. Nokia, Sony Ericsson, Motorola, NTT DOCOMO ja muutama muu yritys ovat perustaneet Symbian Foundation-säätiön, jossa yhdistyvät S60-,

UIQ- ja MOAP-alustat. Yritykset ovat siis lupautuneet antamaan omat ohjelmistoalustansa säätiön käyttöön. Nokia antaa säätiön käyttöön myös omistamansa Symbian-käyttöjärjestelmän. Yritykset kehittävät yhdessä tätä uutta alustaa, johon liitetään ominaisuuksia kaikista mainituista alustoista. Uusi Symbian-alusta on tarkoitus saattaa avoimeksi lähdekoodiksi kahden vuoden kuluessa, jolloin lähdekoodit ovat kaikkien saatavilla. Voidaan siis olettaa, että dokumentaatiosta ja kehittäjäyhteisöistä ei jatkossa tule olemaan puutetta. /13/

5.4 Työn onnistuminen

Työn tavoitteena oli saada testikehys toiminaan UIQ-alustalla kosketusnäyttöä hyödyntäen. Käyttöliittymä piti suunnitella mahdollisimman helppokäyttöiseksi ja esimerkiksi valikoiden liiallista käyttöä tuli välttää. Lisäksi käyttöliittymään tuli tehdä joitakin parannuksia ja lisäominaisuuksia.

Ottaen huomioon sen, että minulla ei ollut yhtään kokemusta UIQ-alustasta, eikä sen enempää kosketusnäytön hyödyntämisestä Symbianiin pohjautuvissa laitteissa ennen tämän työn tekemistä, onnistui työ mielestäni hyvin. Käyttöliittymä UIQ-sovelluksessa on mielestäni helppokäyttöinen ja toimiva.

Lisäominaisuuksista testisettien lataus xml-tiedoston perusteella on mielestäni hyvä lisäys, joka helpottaa ja nopeuttaa sovelluksen käyttöä huomattavasti. Koska ennen ei ollut mahdollisuutta ladata testisettejä, jouduttiin moduulit ensin lataamaan sovellukseen ja sieltä lisäämään halutut testitapaukset testisettiin. Nyt testisetti voidaan määritellä xml-tiedostoon ja sitten vain ladata se. Testisetin latauksen yhteydessä moduulit ja testitapaukset ladataan automaattisesti, käyttäjän tarvitsee vain valita ladattava setti.

5.5 Jatkokehitysajatuksia

Eräs jatkokehitysajatus on testisetin tallennus xml-tiedostoon. Tämän toteuttaminen ei olisi kovin vaativaa, koska xml-tiedoston muoto on jo määritelty. Xml-tiedosto on hyvin yksinkertainen ja tiedot, jotka tiedostoon pitää tallentaa, on helposti

saatavilla. Käytännössä tämän toiminnon lisääminen vaatisi lisäyksiä kontrolleriin, malliin sekä CTestUiXMLHandler-luokkaan, jossa tiedot pitäisi kirjoittaa tiedostoon.

Toisena jatkokehitysajatuksena on kehittää edelleen käyttöliittymän käytännöllisyyttä. Sekä tapaus- että moduulinäkymän listoissa on UIQ-versiossa mahdollisuus valita useita alkioita kerralla. Useiden alkioiden valinta ei tällä hetkellä kuitenkaan vaikuta mihinkään, koska toteutus on tekemättä. Tapausnäkymässä useiden testitapausten valikoiva poisto olisi selkeä parannus ja luultavasti mahdollista toteuttaa melko pienellä työllä. Tällä hetkellähän testitapauksia voi poistaa yksitellen tai kaikki kerralla. Moduulinäkymässä useiden alkioiden valinta olisi myös hyödyllistä joidenkin toimintojen kohdalla. Esimerkiksi usean moduulin poisto kerralla tai kaikkien testitapausten lisäys testisettiin useasta moduulista kerralla parantaisi käyttöliittymän käyttäjäystävällisyyttä entisestään.

LÄHDELUETTELO

Painetut lähteet

5 Harrison, Richard; Shackman, Mark, *Symbian OS C++ for Mobile Phones*, Volume 3. Wiley, 2007.

Sähköiset lähteet

1 UIQ Technology AB, *The basics in Application development tutorial*. UIQ Technology AB, 2007. [www-sivu]. [viitattu 7.10.2008]. Saatavissa: http://developer.uiq.com

2 Nokia, *S60 5th Edition C++ Developer's Library v1.0*. Nokia, 2008. [www-sivu]. [viitattu 13.10.2008]. Saatavissa: http://www.forum.nokia.com

3 UIQ Technology AB, *The UIQ 3 Application Behaviour Guide*. UIQ Technology AB, 2007. [pdf-dokumentti]. [viitattu 20.10.2008]. Saatavissa: http://developer.uiq.com

4 UIQ Technology AB, *Views and dialogs – UIQ Books*. UIQ Technology AB, 2008. [www-sivu]. [viitattu 9.2.2009]. Saatavissa: http://books.uiq.com/index.php

6 Nokia, *S60 5th Edition: What's New For Developers*. Nokia, 2008. [pdf-dokumentti]. [viitattu 11.2.2009]. Saatavissa: http://www.forum.nokia.com

7 UIQ Technology AB, *UIQ releases*. UIQ Technology AB, 2008. [www-sivu]. [viitattu 12.2.2009]. Saatavissa: http://www.uiq.com

8 Symbian Software Limited, *Essential UIQ Getting Started*. Symbian Software Limited, 2006. [pdf-dokumentti]. [viitattu 12.2.2009]. Saatavissa: https://developer.symbian.com

9 Symbian Software Limited, *Front End Processor Overview*. Symbian Software Limited, 2007. [www-sivu]. [viitattu 19.2.2009]. Saatavissa: http://www.symbian.com/

10 UIQ Technology AB, *Understanding UI components – UIQ Books*. UIQ Technology AB, 2008. [www-sivu]. [viitattu 24.2.2009]. Saatavissa: http://books.uiq.com/index.php

11 UIQ Technology AB, *UIQ 3 – "View and Command"*. UIQ Technology AB, 2007. [pdf-dokumentti]. [viitattu 29.2.2008]. Saatavissa: http://developer.uiq.com

12 UIQ Technology AB, *Views and dialogs – UIQ Books*. UIQ Technology AB, 2008. [www-sivu]. [viitattu 25.2.2009]. Saatavissa: http://books.uiq.com/index.php

13 Nokia, *Matkaviestintäalan johtavat yritykset yhtenäistävät ja avaavat Symbianin ohjelmistoalustan*. Nokia, 2008. [www-sivu]. [viitattu 20.4.2009]. Saatavissa: http://www.nokia.fi

APPENDIX
HARRISON20

https://dl.acm.org/doi/10.1145/1434790.1434797 July 30, 2026

ACM DIGITAL LIBRARY | Association for Computing Machinery

Basic Edition Upgrade | Sign in | Register

Journals Magazines Proceedings Books SIGs Conferences Institutions People

Search ACM Digital Library | Advanced Search

Conference Proceedings Upcoming Events Authors Affiliations Award Winners

Home > Conferences > JTRES > Proceedings > JTRES '08 > Enhanced integration of Java to symbian OS using smart pointers

You are using the Basic Edition. Features requiring a subscription appear in grey.

Upgrade

Sign in to your subscription or learn more

RESEARCH-ARTICLE | FREE ACCESS

Enhanced integration of Java to symbian OS using smart pointers

Authors: Raine Mäkeläinen, Cristiano Di Flora, Tommi Mikkonen Authors Info & Claims

JTRES '08: Proceedings of the 6th international workshop on Java technologies for real-time and embedded systems
Pages 38 - 47 • https://doi.org/10.1145/1434790.1434797

Published: 24 September 2008 Publication History Check for updates

0 301 PDF/eReader

Abstract

Porting software from one environment to another can be eased with virtual machines. Then, the virtual machine hides the particularities of one system, and allows running of the same applications in a set of environments. Unfortunately, there are also restrictions related to this approach. In particular, when a program running inside the virtual machine wishes to access the hosting system for e.g. improved performance or service that is readily available in the hosting system but not in the virtual machine, several complications exist. In this paper, we introduce an improved fashion to integrate mobile Java to Symbian OS environment using so-called smart pointers to ease the above problems. Towards the end of the paper, we also provide an evaluation on our prototype implementation to validate the approach.

PDF Help

Formats available

You can view the full content in the following formats:

PDF/eReader

References

[1] Raine Mäkeläinen. *Advanced Java Integration to Symbian OS* MSc. thesis, Department of Information Technology, Tampere University of Technology, Tampere, Finland, 2007.
Google Scholar

[2] Roger Riggs, Antero Taivalsaari, Jim van Peursem, Jyri Huopaniemi, Mark Patel, and Aleksi Uotila *Programming Wireless Devices with the Java#8482;2 Platform, Micro Edition Second Edition* Addison-Wesley Professional, 2003.
Digital Library Google Scholar

[3] Richard Harrison and Mark Shackman *Symbian OS C++ for Mobile Phones, Volume 3* Symbian Press/Wiley, 2007.
Digital Library Google Scholar

[4] Steve Babin *Developing Software for Symbian OS* John Wiley & Sons, Ltd, 2007.

Digital Library Google Scholar

[5] Bjarne Stroustrup *The C++ Programming Language - Third Edition* Addison-Wesley, 1997.
Digital Library Google Scholar

[6] Jo Stichbury *Symbian OS Explained: Effective C++ Programming for Smartphones* John Wiley & Sons, Ltd, 2005.
Google Scholar

[7] Coding Standards, Essential Symbian OS http://developer.symbian.com/main/learning/press/books/pdf/coding_standards.pdf Symbian Software Limited, 2008.
Google Scholar

[8] Liang Sheng *Java#8482; Native Interface: Programmer's Guide and Specification* Addison Wesley Professional, 1999.
Digital Library Google Scholar

[9] Rob Gordon *Essential JNI: Java#8482; Native Interface* Prentice Hall PTR, 1998.
Digital Library Google Scholar

[10] Arvind Gupta and Martin de Jode *Extending the Reach of MIDlets* http://www.arvindgupta.net/pdf/The_Bridge.pdf www.symbian.com, 2005.
Google Scholar

[11] C++ component reference in Symbian OS reference http://www.symbian.com/developer/techlib/v9.1docs/doc_source/reference/referencecpp/index.html
Google Scholar

[12] New IPC Mechanisms for Symbian OS http://developer.symbian.com/main/downloads/papers/newipc/new_ipc_mechanisms_for_symbian_os.pdf {10 May 2008}.
Google Scholar

[13] Hirzel M, Grimm R. Jeannie: Granting Java Native Interface Developers Their Wishes. *ACM SIGPLAN Notices* 2007; **42**(10):19--38.
Digital Library Google Scholar

[14] Bubak M., Kurzyniec D., Luszczek P. Creating Java to native code interfaces with Janet extension. *In Proc SGI Users's Conference* 2000; 283--294.
Google Scholar

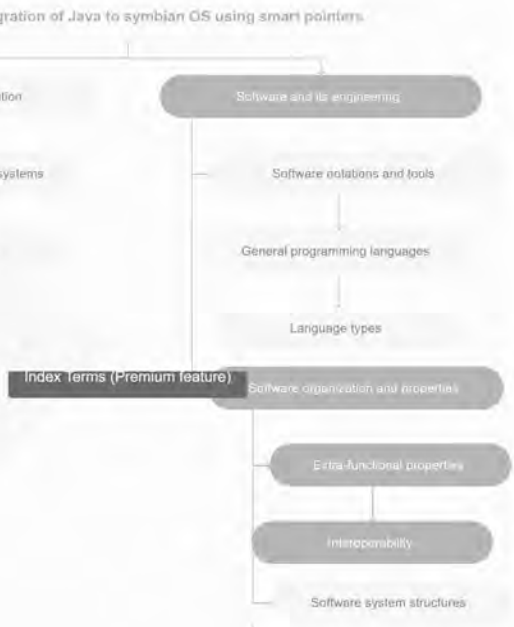
[15] JNIWrapper http://www.teamdev.com/jniwrapper/index.jsf {30 May 2008}.
Google Scholar

[16] Swig http://www.swig.org {30 May 2008}.
Google Scholar

[17] Dr. Dobb's article of NoodleGlue http://www.ddj.com/java/184401984 {30 May 2008}.
Google Scholar

Show fewer references

Index Terms



Embedded software

Real-time systems software

Recommendations

Interactive Cross-Language Pointer Analysis For Resolving Native Code in Java Programs

ICSE '25: Proceedings of the IEEE/ACM 47th International Conference on Software Engineering

Java offers the Java Native Interface (JNI), which allows programs running in the Java Virtual Machine to invoke and be manipulated by native applications and libraries written in other languages, typically C. While JNI mechanism significan...

Read More

State of practices of Java native interface

CASCON '19: Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering

The use of the Java Native Interface (JNI) allows taking advantage of the existing libraries written in different programming languages for code reuse, performance, and security. Despite the importance of JNI in development...

Read More

Using Java APIs with native Ada compilers

SIGAda '96: Proceedings of the 1996 annual ACM SIGAda international conference on Ada

Read More

Comments

DL Comment Policy

Comments should be relevant to the contents of this article, (sign in required).

Got it

0 Comments

• Share

Best Newest Oldest

Nothing in this discussion yet

Privacy Do Not Sell My Data

Download PDF

PDF

Help

View Table of Contents

The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2026 ACM, Inc.
Terms of Usage | Privacy Policy | Code of Ethics

Enhanced Integration of Java to Symbian OS Using Smart Pointers

Raine Mäkeläinen
Nokia Research Center
Tampere, Finland
P.O. Box 1000
raine.makelainen@nokia.com

Cristiano Di Flora
Nokia Innovation Acceleration
Tampere, Finland
P.O. Box 1000
cristiano.di-flora@nokia.com

Tommi Mikkonen
Tampere University of Technology
Tampere, Finland
P.O. Box 553
tommi.mikkonen@tut.fi

ABSTRACT

Porting software from one environment to another can be eased with virtual machines. Then, the virtual machine hides the particularities of one system, and allows running of the same applications in a set of environments. Unfortunately, there are also restrictions related to this approach. In particular, when a program running inside the virtual machine wishes to access the hosting system for e.g. improved performance or service that is readily available in the hosting system but not in the virtual machine, several complications exist. In this paper, we introduce an improved fashion to integrate mobile Java to Symbian OS environment using so-called smart pointers to ease the above problems. Towards the end of the paper, we also provide an evaluation on our prototype implementation to validate the approach.

Keywords

Java Native Interface, Mobile Java integration, Mobile devices programming, J2ME, Symbian OS

1. INTRODUCTION

Porting software from one environment to another can be eased with virtual machines. Then, the virtual machine hides the particularities of one system, and allows running of same applications in different environments. The approach is commonly used, and even restricted systems like mobile phones are capable of hosting other programming environments, most notably mobile Java, which is a commonly available programming environment even in low-performance mobile phones.

Unfortunately, there are also restrictions related to the above approach. One practical problem is that when a program running inside the virtual machine wishes to access the hosting system for e.g. improved performance or service that is readily available in the hosting system but not in the virtual machine, several complications exist. Firstly, composing associated interface functions for both the system running inside the virtual machine as well as the hosting system is often relatively complex, and parameter passing and the relation of objects in native and virtualized system further complicates the system. Moreover, debugging the cooperation of the two systems is commonly considered difficult, which can be harmful in a restricted environment, such as a mobile phone, where debugging and validation are already complicated.

In this paper, a revised version of [1], we introduce an improved fashion to integrate mobile Java [2] to Symbian OS environment [3, 4] using smart pointers – objects that act like pointers and in addition perform some action whenever an object is accessed through them [5].

The paper is structured as follows. Section 2 discusses our motivation and provide background for the paper. Section 3 provides an overview to our approach, and Sections 4 and 5 discuss the most important design decisions. Section 6 provides an evaluation for the composed system, and Section 7 finally draws conclusions. Throughout the paper we will use Symbian specific programming idioms and terms [6, 7].

2. MOTIVATION AND BACKGROUND

The Java Native Interface (JNI) [8, 9] enables two-way integration of native operations and Java programs. Moreover, Java routines that use JNI remain platform independent. Thus, JNI provides considerable added value when the implementation of a Java application requires an enhanced access to enhanced or low-level facilities, or a service already exists in the native implementation, and there is no point in reimplementing a corresponding service in Java. Such Java classes that combine and include native functionality within their business logic are usually referred to as mixed-language classes.

Instead of using JNI, one can also invoke native code from a virtual machine with a socket based integration approach [10], which is often the case in mobile Java devices. However, using such an approach can imply a bigger performance overhead than that of pure JNI invocation, which in turn makes this approach unsuitable for Java applications having strict performance requirements. Therefore, using JNI to improve the capabilities of Java in the device seems to be an ideal choice to take.

One mobile device native platform that offers JNI is Symbian OS [3, 4], which is also our target environment. In this environment there are a lot of existing operations that have been thoroughly tested in product setting, whereas implementing corresponding services in Java would require ad-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
JTRES'08, September 24-26, 2008, Santa Clara, California, USA
Copyright 2008 ACM 978-1-60558-337-2/08/9 ...$5.00.

FIFTH THIRD EX1062
PART 2 of 3
Page 358 of 379

ditional coding, testing, and quality control, as well as additional development time. Moreover, some devices offer JNI capabilities, although only a subset of JNI is usually available. In Symbian OS, native applications are implemented using C++, which is one of the languages that can be used in connection with JNI. This offers an option to use C++ facilities from Java via JNI for improved performance, smaller memory footprint if we assume that the corresponding features would be present in C++ in any case, and faster design, implementation, and testing cycles. Used in devices that offer relatively high performance, Symbian OS contains a number of services that are expected of mobile devices. For instance, services like messaging, telecommunications, personal information management, and device specific capabilities such as Bluetooth might also be valuable when accessed from Java applications. Java Specification Requests (JSR) define how such facilities should be used from Java programs. However, they do not specify to what extent corresponding features should be implemented in Java, but allow the use of native applications where applicable.

Unfortunately, the integration of Java to another environment, such as Symbian OS and its resources for instance, and offering the option to use JNI is not without problems. In particular, interfacing involves numerous details, including Java environment pointers, references to objects through the interface, conversions of objects between the two environments, managing exceptions that may originate from the other side but must be caught and recovered from locally. In addition, issues related to garbage collection must be considered. In short, while the two systems interoperate, they still remain independent and do not share intimate data about their execution automatically. These issues have led us to reconsider how to implement JNI related operations in mobile devices in an improved fashion we refer to as Smart Java Native Invocation Framework (SJNI).

3. SMART JAVA NATIVE INVOCATION FRAMEWORK

3.1 Development approach

The goal of Smart Java Native Interface Framework (SJNI) is to raise the level of abstraction. This is achieved with enhanced tools that generate certain parts of the system. This liberates the developer from performing certain straightforward tasks. However, SJNI does not remove the requirement of understanding the target hosting system, in this case the Symbian environment [6, 4].

Development approaches based on JNI and SJNI are compared in Figure 1. At the beginning of standard JNI integration approach, the programmer writes a Java class that declares native methods (with `native` keyword). These methods are then used to access native services from the hosting system. Next, the programmer compiles Java class files into Java bytecode from which a C++ header file is generated, assuming that Symbian OS is used as the target platform. The generated C++ header file declares same methods that were defined in Java but using C++. Now the programmer can write a C++ source file that implements the generated native C++ header file. Finally, after compiling the native C++ part, a Java-native binding can be used from a Java application.

When using SJNI, the integration is started by defining an SJNI *Wrapper Library Descriptor* (5.2.1). Next, a generator tool provided by SJNI framework is used to create native C++ wrapper library and SJNI *Operation Interface* file (4.3). Finally, the generated native C++ wrapper library is compiled, after which the Java-native binding is available in Java applications.

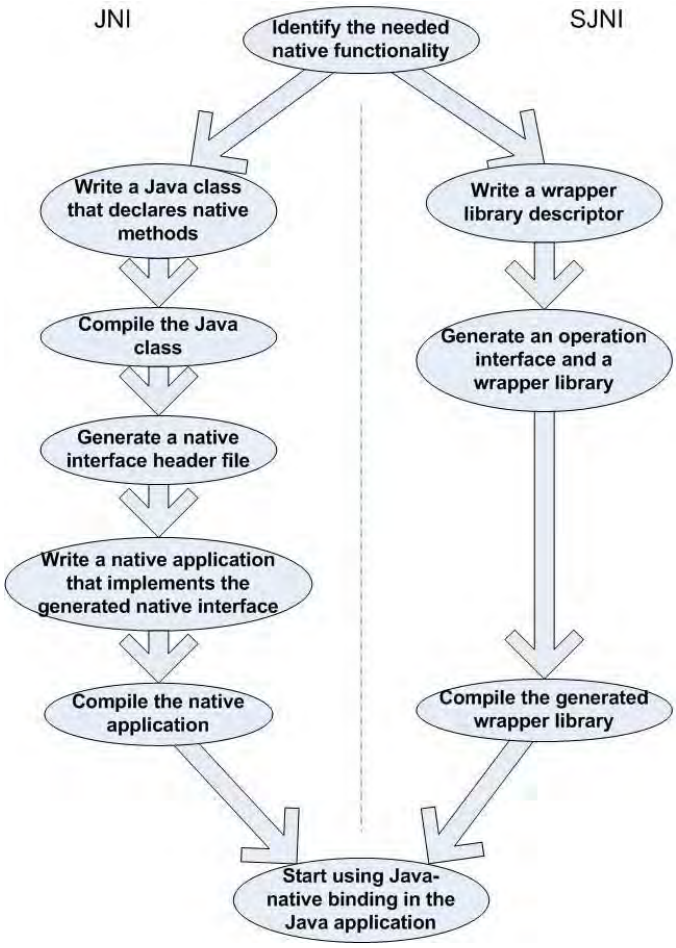


Figure 1: Comparison of JNI and SJNI development approaches

To summarize, the main difference between the approaches is that when using SJNI integration one does not have to deal with regular JNI tools, but a generator will be used instead. In addition, an SJNI developer do not need to understand the basics of JNI development, and he can thus concentrate on his own application business logic.

3.2 Architecture

The architecture of the Smart Java Native Invocation (SJNI) Framework that supports the approach described above consists of three main sub-systems, namely *Java Stub*, *Java Smart Pointer*, and *Invoker*. The overall architecture of the framework is outlined in Figure 2, where the *Java Application* and *Native Service* sub–systems are not parts of the SJNI framework. The former represents a mixed-language Java component, whereas the latter represents the native APIs it needs to invoke associated operations. As far as the SJNI framework is concerned, the role of the three

FIFTH THIRD EX1062
PART 2 of 3
Page 359 of 379

key subsystems can be summarized as follows.

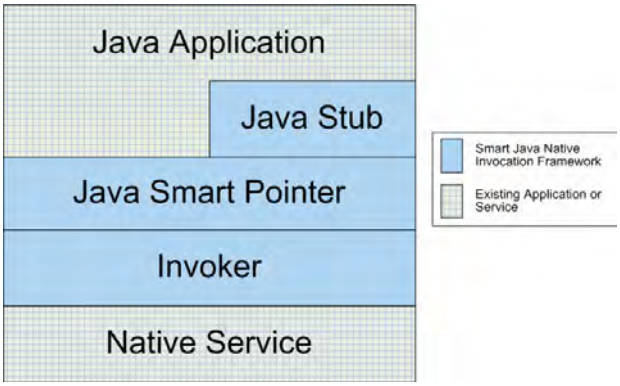


Figure 2: Overall layer architecture of a mixed-language Java application implemented through the Smart Java Native Invocation Framework

Java Stub sub-system consists of one or more Java components (so-called SJNI stubs) each of which implements a *Native Service* specific Java interface. Each *SJNI stub* consists of a thin wrapper built on top of the *Java Smart Pointer* and provides a type safe interface for invoking functionality of native services.

Java Smart Pointer (*JSP*) represents a Java-side implementation of the well-known smart pointer concept. The JSP provides Java applications with a Java-side pointer to a native entity (object, structure, variable) allocated on native heap. The pointer is "smart" because it performs additional steps when a native object is accessed or created. For instance, after the construction of a JSP object, the JSP instance is responsible for lifecycle management of the related native entity as well as for loading and unloading the necessary native DLLs.

Invoker is responsible for the actual invocation of the target C++ *Native Service*. It provides a generic type-unsafe interface for dynamic native library loading, invocation of C/C++ functions and methods, as well as for dynamic creation and destruction of C/C++ variables (objects, structures, variables) on the native C++ heap. The *Invoker* is a private component that can be only accessed by the *Java Smart Pointer* subsystem. It is never invoked explicitly by *Java Application* logic, which on the contrary must use either the type-unsafe *Java Smart Pointer* or a type-safe *Java Stub* in order to access the target *Native Service*.

Figure 3 shows the main interfaces implemented by the Smart Java Native Invocation Framework components. The UML stereotypes adopted in Figure 3 are in the following. Stereotype *JNI* is a mixed-language component that has Java native interface defined at Java side and native implementation written with Symbian C++ at native side, stereotype *Native Class* stands for a class that is written with Symbian C++, stereotype *Class Metadata* describes metadata for a native class, and stereotype *DLL* represents an external library. A technical restriction we had to take into account throughout the design is that already existing DLLs were to be used, as we did not want to increase native side static footprint when methods or functions were exported by a DLL [7].

In the following sections we will provide a detailed description of how we designed and implemented the above

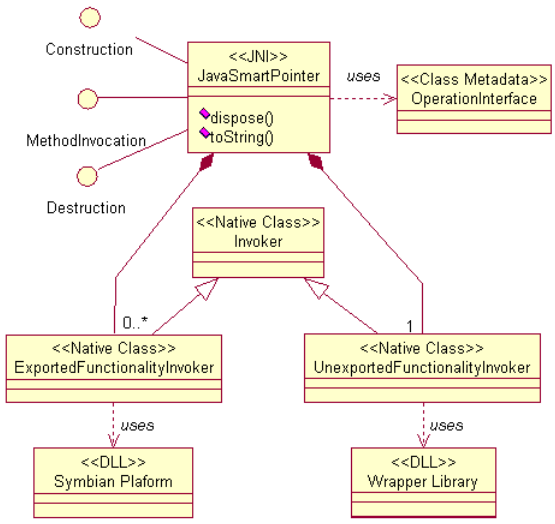


Figure 3: UML class diagram of JSP

components. Section 4 addresses the Java Smart Pointer component, and Section 5 discusses *SNJI Wrapper Library Descriptor*, code generation, and the invokers used in the integration.

4. JAVA SMART POINTER COMPONENT

4.1 Overview

The *Java Smart Pointer* (JSP) has been implemented as a mixed-language component using standard JNI coding. However, the JSP interface does not contain any JNI-dependent features, and the SJNI programmer is not supposed to be familiar with JNI in order to start using the JSP.

The *JSP* allows creating, invoking, and destroying native entities through three different interfaces, referred in the following as *Construction*, *MethodInvocation*, and *Destruction* interfaces. Furthermore, each *JSP* instance can contain different types of SJNI *Invokers*, namely the *ExportedFunctionalityInvoker* (EFI) and the *UnexportedFunctionalityInvoker* (UFI), designed to invoke native APIs that can be or cannot be exported by a DLL, respectively.

4.2 Representing C++ pointers as SJNI components

The main idea behind the *Java Smart Pointer* component is to dynamically create new instances of a certain native classes or variables, and store them on the heap for further usage (method or function invocation). To this aim we had to explicitly take into account the problem of representing all the main idioms related to standard C++ programming as well as to Symbian C++ dialect [6, 7], when designing the SJNI framework. We also had to carefully design and implement support for SJNI-specific concepts.

Since the main goal of the SJNI framework is to enable Java consumers to create, invoke, and destroy native C++ objects at run-time, the first problem we faced was the design of the Java representation of C++ object pointers. The *Java Smart Pointer* can thus be considered as a

FIFTH THIRD EX1062
PART 2 of 3
Page 360 of 379

Java-side wrapper for a generic C++ object pointer (i.e., a void pointer or a `TAny` pointer according to Symbian terminology), enabling dynamic creation, invocation, and destruction of native objects. In addition to object pointers, we had to represent C++ function pointers as well as pointers to C++ structures, since a lot of native APIs use such pointers as method arguments or as values returned by methods/functions. These types of C++ pointers are represented in the SJNI framework by three different Java classes, `JSPObjectPointer`, `JSPFunctionPointer`, and `JSPCStruct`. All pointer classes are derived from the same abstract class `JavaSmartPointer`, which represents the common logic associated to SJNI representation of native pointers. Thus it is possible to create native pointers (e.g., Object, structure, or function pointers), wrap them into an instance of the related Java-side peers (e.g., `JSPObjectPointer`, `JSPCStruct`, or `JSPFunctionPointer`), and reuse them for further processing (e.g. passing an object pointer and a function pointer as parameters to another object's method).

4.3 Operation Interface

SJNI *Operation Interfaces* are simple Java enumerations (actually implemented as Java interfaces since the J2ME runtime lacks the concept of enumeration) that are associated to a specific native class that has to be made available to the Java-side by means of the SJNI framework. Operation interfaces are used to make low-level SJNI invocation (i.e. using a JavaSmartPointer) friendlier to developers without compromising its flexibility. They contain all the pieces of information that are needed for creating, invoking, and destroying native objects through an instance of the JavaSmartPointer class. In other words, an SJNI operation interface consists of an enumeration of class-specific constant variables that Java consumers must use in order to create, invoke, and destroy native objects at run-time.

Low-level creation, invocation, and destruction of native objects will be described in the following subsections. However before detailing the design of such low-level invocation primitives, the main structure and semantics of SJNI operation interfaces must be described. Such operation interfaces include the following fields:

- `LIBRARYNAME` is a string array describing the native DLLs that must be loaded at run-time in order to invoke the target functionality (libraries implementing the target native API and its own dependencies);
- `OBJECT_SIZE` represents the size of a native target object instance. This size is needed when the native instance is constructed though EFI.
- `NEED_EFI_INVOKER` indicates whether an EFI is needed for this class or not.
- `NEED_UFI_INVOKER` indicates whether an UFI is needed for this class or not.
- `METHOD_DESCRIPTOR` describes how the native target method is invoked. The name of this field is derived from the native method's signature. The value of this field is an integer array that contains wrapper library index or ordinal of the method. In addition, it contains an indicator that tells which Invoker (EFI or UFI) to use for this method. Method descriptor for constructors starts with native class name and it is followed by arguments for native constructor.

Metadata is used at runtime by the JSP to construct or invoke objects based on SJNI calls requested by a Java SJNI-enabled application. For example, JSP's constructors can load operation interface during construction and check how to load and retrieve native object's constructors and methods from the native DLL that includes them (indicated by field `LIBRARYNAME`). When invoking a method, its description is retrieved from the operation interface.

In the following we show an example of the SJNI Operation Interface generated from the native Symbian C++ `TFullName` class definition:

```
public interface TFullName_Operations
    extends TDes16_Operations
{
public static final boolean NEED_EFI_INVOKER =
    true;
public static final boolean NEED_UFI_INVOKER =
    true;

// Has inlined constructors
public static final int OBJECT_SIZE  = 0;
public static final String [] LIBRARYNAME =
    {"euser.dll"};
public static final int[] TFULLNAME_DELETE_ =
    {0, USE_WRAPPER};
public static final int[] LENGTH_    =
    {1, USE_WRAPPER};
public static final int[] SIZE_      =
    {2, USE_WRAPPER};
public static final int[] MAXLENGTH_ =
    {3, USE_WRAPPER};
public static final int[] MAXSIZE_   =
    {4, USE_WRAPPER};
public static final int[] TFULLNAME_ =
    {5, USE_WRAPPER};
public static final int[] TFULLNAME_TInt =
    {6, USE_WRAPPER};
public static final int[]
    TFULLNAME_TText_const_PTR_ = {7, USE_WRAPPER};
public static final int[]
    TFULLNAME_TDesC_const_BYREF_
        = {8, USE_WRAPPER};
}
```

These metadata can be interpreted as follows. First, the declaration of the class includes information about the native class name (`TFullName_Operations`) and its relationships with other classes (`extends TDes16_Operations`). Since the actual implementation of the `TFullName` class includes both inlined functions and functions exported from a DLL, both the two related flags (`NEED_XFI_INVOKER`) are set to `true`. `OBJECT_SIZE` attribute is set to zero because all constructors are inlined and wrapped, and the `OBJECT_SIZE` parameter is useful only when native constructors are implemented as operations that are exported from a DLL. SJNI operations interfaces should not be hand-coded by Java developers since all the pieces of information that they contain can be retrieved automatically.

4.4 Construction Interface

Native objects can be constructed through the JSP construction interface, which consists of several constructors

FIFTH THIRD EX1062
PART 2 of 3
Page 361 of 379

supporting different prototypes of native constructor definitions. For example, in the following we show a method of the JSP construction interface that should be used to create native objects having one-Int-argument constructors:

```
/**
 *  Constructs a native instance with one
 *  argument of type int.
 */
public JavaSmartPointer(
    String nativeClassName,
    int[] constructorDescriptor,
    int arg0
);
```

This example code shows that in order to construct a native object instance by using an SJNI JSP instance, the following data about the native API is needed:

- Native class name, `nativeClassName`, defines the name of target native class. With `nativeClassName` JSP can retrieve the Operation Interface and load the wrapper library for UFI.
- Method descriptor, `constructorDescriptor`, defines the constructor that is invoked. The method descriptor is defined in Operation Interface.
- JSP needs similar arguments than native target constructor so that instances can be constructed properly.

Based on the above data, a JSP instance can create a native target object instance on the native heap to support further use of that object through the JSP invocation interface. During native-object construction phase, the *Java Smart Pointer* must create and initialize one or more Invokers, based on the actual contents of the Operation Interface related to the native object to construct. Each Invoker, disregarding if it is an EFI or an UFI instance, must load a native DLL during its initialization process. The library can either be a standard Symbian OS native DLL (e.g. `euser.dll`) in the case of EFI initialization, or an SJNI wrapper library for a specific unexported native API in the case of UFI initialization.

The number of initialized Invokers depends on the number of native libraries needed to create an instance of a certain native object. Native Symbian library names are retrieved from Operation Interface and assigned to dedicated Exported Functionality Invoker instances (one EFI instance for each library to load). A native C++ inheritance hierarchy of a class might include multiple libraries that must be loaded when an instance of a native object is created.

Based on the second integer value, defined by `constructorDescriptor`, the constructor is invoked through EFI or UFI. When the construction goes through EFI, then also the UFI is initialized and the library is assigned to it based on `nativeClassName`. The reason why the number of UFIs is limited to one is that `nativeClassName` is mapped to one wrapper library. If construction is called through UFI, then the native target instance is constructed through it. In addition, if EFI (defined in Operation Interface with tag `NEED_EFI_INVOKER`) is needed, then as many Exported Functionality Invoker are initialized as there are libraries defined in Operation Interface (tag `LIBRARYNAME`). Each EFI dynamically loads the library that is assigned to it, but it does not construct any native instances, since UFI has already constructed the native target instance. JSP takes care that each constructed library has the native pointer after construction.

4.5 Method Invocation Interface

The following code snippets describe a set of methods for invoking different kind of native methods that are provided by JSP.

```
/**
 * Invokes a native method without arguments. Native
 * method has no return value.
 */
public void invokeVoidNoArgs(
    int[] methodDescriptor )
    throws SymbianLeave;

/**
 * Invokes a native method with two int arguments.
 * Native method has no return value.
 */
public void invokeVoidTwoArgs(
    int[] methodDescriptor,
    int arg0,
    int arg1 )
    throws SymbianLeave;

/**
 * Invokes a native method with one JavaSmartPointer
 * reference. Native method returns int.
 */
public int invokeIntOneArg(
    int[] methodDescriptor,
    JavaSmartPointer refObject )
    throws SymbianLeave;
```

The example code shows that in order to invoke a native method by using an SJNI JSP instance, the following data about the native API is needed:

- Parameter `methodDescriptor`, defines the method that is invoked. The method descriptor is defined in Operation Interface.
- Similar arguments are needed for JSP method as are needed for native target method so that method can be invoked properly.

A native method can either be a method that is exported from some DLL or method that is not exported by any DLL. Hence, `methodDescriptor` can contain an ordinal of the method in the DLL so the EFI can invoke the method. Alternatively, `methodDescriptor` can contain an index of the method in wrapper library so that the UFI can invoke the method. JSP figures out whether the method is invoked through EFI or UFI based on information provided in Operation Interface.

Furthermore, multiple native Symbian libraries might get loaded during construction of JSP. Hence, JSP needs to know which Invoker is responsible for the required library. The Invoker is figured out based on the second integer value

FIFTH THIRD EX1062
PART 2 of 3
Page 362 of 379

defined in `methodDescriptor`. After the correct Invoker is found, the method can be invoked with the given arguments.

In the native JSP implementation most of native method invocations are surrounded with `TRAP` macros, which are used to catch exceptions in Symbian environment. Then, if a Symbian exception occurs Java exception `SymbianLeave` is thrown.

4.6 Destruction Interface

The JSP takes care of the native object disposing and the disposing of the allocated Invokers. The Java Smart Pointer disposes the allocated native instance by invoking wrapped delete method from assigned wrapper library. Delete is defined in the wrapper library at index zero.

```
public void dispose();
```

4.7 Marshalling arguments

The native implementation of the JSP is responsible for marshalling Java arguments to native ones. Java primitive data types are converted into native primitive data types by the JNI. The native side of the JSP is responsible for marshalling Java objects into native primitives or instances, for example following conversations are available.

- `java.lang.Integer` is mapped to `TInt`.
- `java.lang.Double` is mapped to `TReal`.
- `java.lang.String` is mapped to `TDesC`.
- `JavaSmartPointer` is mapped to `void*` of native object instance. This is possible because native instances are always allocated from the heap and the Invoker is aware of the native pointer.

Arguments of native method are always converted to opaque pointer (`void*`) arguments. The native side of JSP is responsible for such a pointer-casting feature. Since the length of a pointer is 32 bits, we need to take a reference out of those values that are longer than 32 bits. After that, the pointer can be casted it into opaque pointer. The generic Invoker class is realized on top of opaque pointer. Opaque pointer allows invocation of different kind of constructors, a set of methods and the destructor for allocated instance. The native side of the JSP delegates the responsibility of invocation to Invoker that executes the needed functionality.

4.8 Constructing instances of native objects

JSP is a composite of multiple Invoker instances. Therefore, the responsibility of constructing native object instance must be clear. The native target object instance is constructed through the Invoker that has loaded the correct library for object construction. JSP class is responsible for figuring out the correct library owner.

Unexported Functionality Invoker based construction is depicted in Figure 4. First, JSP finds out the wrapper library name based on the `nativeClassName` with `GetWrapperLibrary` function. In the `NewL` function call UFI instance is constructed. During the construction UFI loads the `wrapperLibrary` and calls the function at `constructorIndex` from its wrapper library function table. Now the JSP can get the opaque pointer `(void*)` of the constructed instance by calling `GetNativeInstance`. Finally the JSP constructs as many Exported Functionality Invoker instances as there are native libraries assigned for JSP, and each constructed EFI has the same opaque pointer to the native target instance as UFI. Details of the internals of constructor of Unexported Functionality Invoker are described in 5.2.

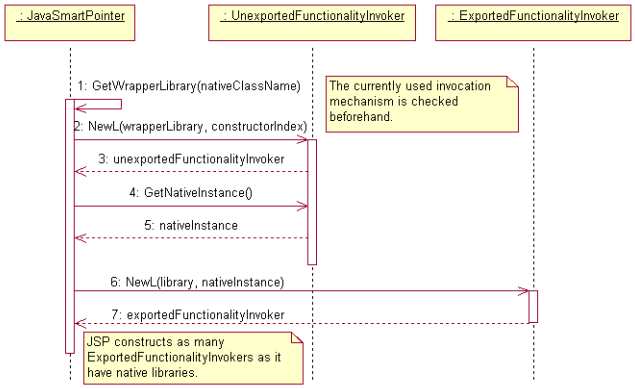


Figure 4: Unexported Functionality Invoker based construction

Let us consider the construction of a native `TFullName` instance. Class `TFullName` has an inlined constructor and it uses `euser.dll`. Since it has an inlined constructor, the constructor must be wrapped inside wrapper library. The steps in Figure 4 apply for the class `TFullName`.

- `GetWrapperLibrary` returns a `wl_tfullname.dll` descriptor.
- `NewL` call to Unexported Functionality Invoker loads the `wl_tfullname.dll` and constructs the native instance.
- After the construction, it is possible to retrieve the opaque `void` pointer to the native instance by calling `GetNativeInstance()`.
- Finally Exported Functionality Invoker can be constructed with library `euser.dll` and opaque void pointer of the native instance.

If the constructor is exported by native library the sequence diagram of that case is depicted in Figure 5. Internal construction of Exported Functionality Invoker is described in 5.3.

5. INVOKER COMPONENTS

5.1 Overview

In order to better understand the role of *ExportedFunctionalityInvoker* (EFI) and *UnexportedFunctionalityInvoker* (UFI) components, let us consider how native C++ APIs are actually processed by C++ compilers/linkers during the build process. According to the standard C++ compilation and linking process, the object code of native APIs exported by some DLL does not need to be included into the object code of other native modules. Consequently, we can create a specific invoker type, namely the EFI, that allows invoking a large number of operations via exported native APIs

FIFTH THIRD EX1062
PART 2 of 3
Page 363 of 379

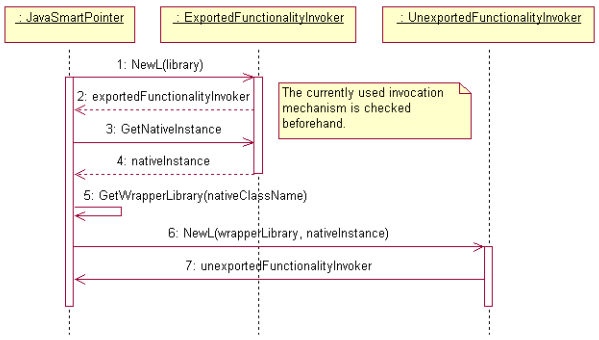


Figure 5: Exported Functionality Invoker based construction

by using the same core native C++ implementation of the EFI. In other words, the EFI native core does not depend at all from the target native APIs to invoke, and allowed us to enable access to a wide range of native APIs with minimum impact on the static memory footprint of the final SJNI binaries.

However, having only EFI-type Invokers in the SJNI framework would have reduced its scope significantly, since native APIs typically include also other C/C++ definitions and declarations, like template and inline methods/functions. These can pose several build-time API-specific dependencies between the target APIs of the native platform and the native implementation of a Java consumer component. These dependencies are due to the fact that C/C++ compilers generate templates instantiations and expand inline code only at the calling block and add them statically to the calling component's object code at build-time. Hence, in addition to DLL-based invocation, the SJNI framework provides also helper components for invoking functionality that does not have any corresponding entry points in DLLs and thus may pose some build-time dependencies to the native part of the overall mixed-language Java application (e.g., implemented as inlined, inherited, and/or template logic). The SJNI therefore provides the so-called UFI invoker type, which assumes that the functionality to import has been wrapped within a native-service specific *Wrapper Library* generated automatically from native source tree by using the SJNI helper tools. Such libraries must be created, compiled, and deployed onto the target device (thus increasing the static memory footprint anytime a new wrapper library is needed).

By using both the EFI and UFI invocation strategies the SJNI framework can still enable access to methods that are not exported by any DLL, but at the same time the memory footprint is minimized since a large number of classes and methods can be invoked by just using the small-size EFI core logic.

5.2 UnExported Functionality Invoker

5.2.1 Wrapper Library Descriptor

Unexported functions must be wrapped inside a library so that Unexported Functionality Invoker can invoke them. The wrapper library is loaded at runtime when an instance of UFI is constructed. The generator tool of the framework generates a wrapper library based on a SJNI *Wrapper Library Descriptor*.

SJNI *Wrapper Library Descriptors* are plain text files including a few key meta-data about the methods and functions to wrap. The generator tool of the JSNI framework generates a wrapper library based on a wrapper library descriptor. When several template and inline methods/functions, which can pose several build-time dependencies, need to be integrated and bound to the Java runtime, metadata need to be defined in the so-called SJNI *Wrapper Library Descriptor*. Meta-data included in an SJNI *Wrapper Library Descriptor* includes the following definitions:

- Wrapped class is defined with `!Class` tag.
- Header file location of wrapped class is defined with `!Location` tag.
- Library to which wrapped class is linked against is defined with `!Linkagainst` tag.
- Inheritance of wrapped class is defined with `!Inheritance` tag.
- Unexported constructors are defined with `!Constructor` tags.
- Unexported methods are defined with `!Method` tags.

For instance, the `TFullName` class is type-defined as `TBuf<512>` class that has inlined constructors. Hence, constructors of `TFullName` class must be declared in SJNI *Wrapper Library Descriptor* file. Also all inline or template methods that `TFullName` class inherits throughout inheritance hierarchy must be declared.

An example *Wrapper Library Descriptor* for the `TFullName`class can be defined as follows:

```
!Class TFullName
!Location       e32cmn.h
!Linkagainst euser.lib
!Inheritance TDes16
!Constructor inline TFullName()
!Constructor inline TFullName(TInt aLength)
!Constructor inline TFullName(const TText *aString)
!Constructor inline TFullName(const TDesC &aDes)
!Method delete TFullName()
!Method inline TInt Length() const
!Method inline TInt Size() const
!Method inline TInt MaxLength() const
!Method inline TInt MaxSize() const
```

5.2.2 Creation of Wrapper Libraries

The generator creates the Wrapper Library header file and the source file based on the wrapper library description. The header file of `TFullName` wrapper library, namely `wl_tfullname.h`, is shown below. In addition, a source file is generated. The source file only invokes the method, which is not described in this context. The library exports a function `GetWrapperLibrary` that returns `WRAPPER_LIBRARY` pointer. `WRAPPER_LIBRARY` contains an array of function pointers to wrapped functions (`WL`) and a size of the array (`WL_size`).

```
#ifndef WL_TFULLNAME_H_HEADER
#define WL_TFULLNAME_H_HEADER
#include "wl.h"
```


FIFTH THIRD EX1062
PART 2 of 3
Page 364 of 379

```
#include <e32cmn.h>

IMPORT_C const
    WRAPPER_LIBRARY * GetWrapperLibrary();

void   WL_TFullName_delete_  (
    TFullName * aThisObject
    );
TInt   WL_TFullName_Length_  (
    TFullName * aThisObject
    );
TInt   WL_TFullName_Size_    (
    TFullName * aThisObject
    );
TInt   WL_TFullName_MaxLength_(
    TFullName * aThisObject
    );
TInt   WL_TFullName_MaxSize_ (
    TFullName * aThisObject
    );
void * WL_TFullName_          ();
void * WL_TFullName_TInt      (TInt aLength);
void * WL_TFullName_TText_const_PTR_   (
    const TText *aString
    );
void * WL_TFullName_TDesC_const_BYREF_ (
    const TDesC &aDes
    );

// Wrapper library function table
const unsigned int WL []= {
   (unsigned int) WL_TFullName_delete_,
   (unsigned int) WL_TFullName_Length_,
   (unsigned int) WL_TFullName_Size_,
   (unsigned int) WL_TFullName_MaxLength_,
   (unsigned int) WL_TFullName_MaxSize_,
   (unsigned int) WL_TFullName_,
   (unsigned int) WL_TFullName_TInt,
   (unsigned int) WL_TFullName_TText_const_PTR_,
   (unsigned int) WL_TFullName_TDesC_const_BYREF_
};
const int WL_size = 9;

#endif
```

The construction of an Unexported Functionality Invoker instance goes through the following steps (the last step also apply to method invocation):

1. Unexported Functionality Invoker loads the given wrapper library at runtime by using `RLibrary` class.
2. Unexported Functionality Invoker gets the wrapper library function table from the loaded library by using `Lookup` function of `RLibrary`. The wrapper library function table is stored into the member variable for further use.
3. Finally, Unexported Functionality Invoker invokes the function of a given wrapper library function table index with the given arguments.

5.3 Exported Functionality Invoker

Exported Functionality Invoker uses native libraries that are provided by Symbian platform. The construction of an Exported Functionality Invoker instance goes through the following steps (the last two steps also apply to method invocation):

1. Exported Functionality Invoker loads a given library at runtime by using `RLibrary` class.
2. Exported Functionality Invoker allocates a memory space for the native instance by calling constructor of `CBase` class with given size.
3. Exported Functionality Invoker retrieves a function from the loaded library by using `Lookup` function of `RLibrary` class.
4. Exported Functionality Invoker invokes the function that is has retrieved with the given arguments.

6. ANALYSIS

6.1 Static footprint

The Smart Java Native Invocation Framework implementation contains a set of Java classes and a native core containing Java Smart Pointer, Invoker, and Utils components. Utils provides C++ helper classes for Java Smart Pointer and Invoker. Java classes and native core components do not contain any generated code such as wrapper libraries or operation interfaces. Compiled Java classes that are addressed in this subsection do not contain any debug information.

The overall Static footprint of the framework is around 35 KB. Figure 6 shows the breakdown of this footprint among the different main components of the SJNI implementation. Java side static footprint accounts for around 43% of the framework size, and the remaining 57% comes from native core libraries. Thus, the framework fits easily in the restricted memory of devices such as mobile phones.

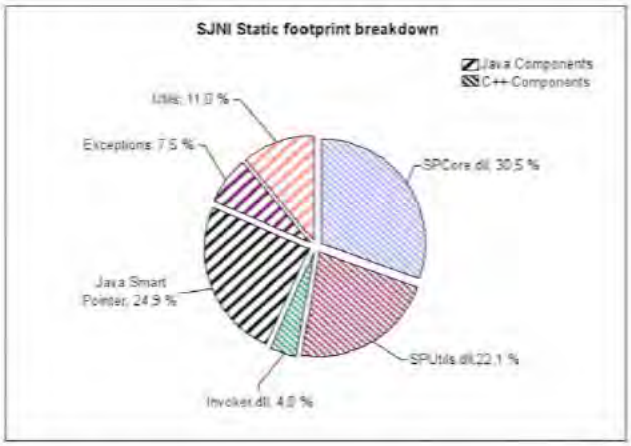


Figure 6: Breakdown of the overall static footprint of Smart Java Native Invocation Framework

As far as the native C++ libraries are concerned, the Java Smart Pointer (`SPCore.dll`) is responsible for 30.5% (i.e., 10.5 KB), Invoker (including EFI and UFI) is responsible for around 4% (i.e., 1.4 KB) and different helper Utils are responsible for around 22% (i.e., 7.7 KB). Java side static footprint of the framework is distributed in the following fashion: Java Smart Pointer is responsible for 24.9% (i.e., 8.7

FIFTH THIRD EX1062
PART 2 of 3
Page 365 of 379

KB), Exceptions are responsible for 7.5% (i.e., 2.6 KB), and the remaining 11% (i.e., 3.8 KB) contains different helper Utilities used by other SJNI Java components.

JSP's way to invoke DLL exported methods of native services does not increase native size of JSP. However, in JNI there is the need to a write function call to native side. Furthermore, developer does not see how the real method invocation is done inside JSP, JNI is invisible to the developer.

In order to give the reader an idea of how many native APIs can be invoked without adding any wrapper library (and thus without increasing the static footprint), one can consider that in Symbian OS public C++ APIs [11] about 12800 methods are exported by some system dynamic library and thus can be invoked through SJNI without writing wrappers, whereas about 4950 methods are not exported by any dll and thus must be wrapped. To summarize, we estimated that about 72% of all Symbian OS native API methods can be invoked directly without writing a wrapper, which implies that a lot of native APIs can be invoked from Java without adding any extra contribution to the 35KB static footprint of the SJNI implementation.

6.2 Development efficiency

Based on our experiences and experiments, writing a Java stub for a native target class on top of JSP reduces the lines of code to be written compared to Java stub written with JNI. The number of lines is reduced because JSP provides a set of functions to invoke. In addition, there is no need to write native method invocation that is necessary in JNI case. Furthermore, wrapper is generated based on wrapper library descriptor that eases wrapper library creation.

In Figure 7 there are two native classes, `TFullName` and `RMsgQueue<TInt>`, integrated from native side to Java side. Class `TFullName` is a Symbian descriptor class and `RMsgQueue<TInt>` is a template class that represents one of the available Inter-Process Communication (IPC) mechanisms of Symbian OS [12].

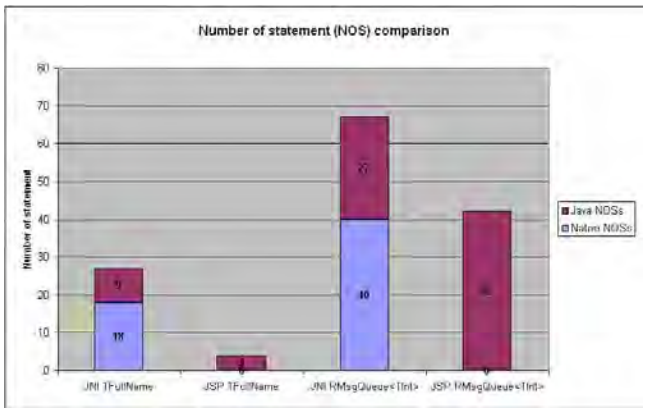


Figure 7: Number of statements (NOS) comparison

In order to compare development efficiency of JNI and SJNI coding, we implemented Java-side representations of both the classes and evaluated the number of statements (NOS) in each implementation. The number of statements rather than the lines of code (LOC) is used, since multiple statements might be located at the same line. From `TFullName` class only six methods are integrated (including two constructors), whereas twelve methods from `RMsgQueue<TInt>` class (including one constructor) are integrated. In the rest of this section, we will mostly focus on the `RMsgQueue<TInt>` class, since only a few methods needed to be integrated for `TFullName`. A full list of the integrated methods of `TFullname` and `RMsgQueue<TInt>` classes is reported in Figures 8 and 9, respectively.

SJNI implementation of `RMsgQueue<TInt>` required almost the same number of Java-side statements that the JNI implementation required on the native side. This illustrates the idea of invoking native target method directly from Java. In JNI implementation, we had to write native side code in which we invoked the native target method, as well as Java-side code where we had to declare native methods as private and public methods where we invoke native methods. Moreover, in public methods we had to add a lot of JNI boilerplate code in order to represent native-side pointers and integer Java variables. JNI Java side implementation increases the number of statements to be written. To conclude, in SJNI we did not have to write any additional Java code but we could directly invoke the target netive method from Java. In this case, the JNI implementation required us to write additional 25 statements (37%) that were not required at all in the SJNI implementation.

	JNI TFullName		JSP TFullName	
Method	**Java NOS**	**Native NOS**	**Java NOS**	**Native NOS**
TFullName()	2		1	
TFullName(java.lang.String)	2		1	
append(java.lang.String)	1		1	
delete(int, int)	1		1	
dispose()	1		0	
toString()	2		0	
_constructor()		3		
_constructor(java.lang.String)		4		
_append(int, java.lang.String)		3		
_delete(int, int, int)		2		
_dispose(int)		3		
_toString(int)		3		
	9	**18**	**4**	**0**

Figure 8: Integrated methods of TFullName and NOS for each method

	JNI RMsgQueue<TInt>		JSP RMsgQueue<TInt>	
Method	**Java NOS**	**Native NOS**	**Java NOS**	**Native NOS**
RMsgQueue_TInt()	2		1	
createGlobal(java.lang.String, int)	2		2	
createGlobal(java.lang.String, int, int)	2		2	
createLocal(int)	2		2	
createLocal(int, int)	2		2	
receive()	5		11	
receiveBlocking()	2		5	
send(int)	4		8	
sendBlocking(int)	1		3	
messageSize()	2		2	
close()	1		4	
dispose()	2		0	
_close(int)		2		
_craeteLocal(int, int, int)		3		
_createGlobal(int, java.lang.String, int)		4		
_createGlobal(int, java.lang.String, int, int)		4		
_createLocal(int, int)		3		
_dispose(int)		2		
_messageSize(int)		3		
_receive(int)		7		
_receiveBlocking(int)		4		
_RMsgQueue_TInt()		3		
_send(int, int)		3		
_sendBlocking(int, int)		2		
	27	**40**	**42**	**0**

Figure 9: Integrated methods of RMsgQueue<TInt> and NOS for each method

Since JSP is a mixed-language component that enables native integration via JNI without writing error-prone JNI code at all, it eases the development. Developers do not

FIFTH THIRD EX1062
PART 2 of 3
Page 366 of 379

need to perform the time consuming JNI testing and debugging phase, as this is offered by the framework. In addition, SJNI maps native exceptions – in the case of Symbian, so-called leave codes – into Java exceptions, which provides the developer with a much more Java friendly environment to work with.

An additional advantage is that programmers can combine several native method invocations inside one procedural JNI invocation. This naturally reduces the overhead caused by the JNI invocation, because Java Virtual Machine needs to handle only one invocation. The downside of combining multiple native method invocations into a single JNI invocation is that we are compromising reusability, because then the combined invocation does one atomic operation and pieces of that code cannot be used separately.

7. CONCLUSIONS

In this paper, we have introduced a mechanism called Smart Java Native Integration Framework (SJNI) for enhanced integration of native and Java code in mobile devices. The Smart Java Native Integration Framework allows Java programmer to take native side service into use in Java without writing JNI glue code by themselves, which in turn helps in reducing the static-footprint of complex mixed-language Java components and simplifies their development tasks.

The static footprint analysis showed that the framework requires about 35KB of Java and native code to be deployed onto the mobile device, which means that the framework has a minimal static footprint impact. The number of manual actions needed when implementing integration between Java and native side compared to same integration with JNI is reduced when SJNI is used. Since JNI native function is a counterpart for a method declared at Java side, we had to bind Java methods and native functions. We showed that the binding is generic enough to allow the invocation of different methods. We showed that around 72% of native Symbian OS APIs can be invoked without adding any additional native wrapper library.

We have found multiple JNI helper frameworks or tools [13, 14, 15, 16, 17] that are more or less based on compilers and code generators. These tools also aim at lowering the entry barrier for starting to develop JNI-enabled Java software. However, all these tools increase also the static-footprint, since there is new native code introduced. Moreover, it is very likely that all of these tools are not applicable on mobile environement that has its own specialities such as string descriptors in Symbian C++.

In the future, we aim at gaining more experiences on using the framework in practice. In particular, feedback from developers that have not participated in the development of the framework is considered important and will be thoroughly evaluated.

8. REFERENCES

[1] Raine Mäkeläinen. *Advanced Java Integration to Symbian OS* MSc. thesis, Department of Information Technology, Tampere University of Technology, Tampere, Finland, 2007.

[2] Roger Riggs, Antero Taivalsaari, Jim van Peursem, Jyri Huopaniemi, Mark Patel, and Aleksi Uotila *Programming Wireless Devices with the Java^{TM}2 Platform, Micro Edition Second Edition* Addison-Wesley Professional, 2003.

[3] Richard Harrison and Mark Shackman *Symbian OS C++ for Mobile Phones, Volume 3* Symbian Press/Wiley, 2007.

[4] Steve Babin *Developing Software for Symbian OS* John Wiley & Sons, Ltd, 2007.

[5] Bjarne Stroustrup *The C++ Programming Language - Third Edition* Addison-Wesley, 1997.

[6] Jo Stichbury *Symbian OS Explained: Effective C++ Programming for Smartphones* John Wiley & Sons, Ltd, 2005.

[7] Coding Standards, Essential Symbian OS http://developer.symbian.com/main/learning/press/books/pdf/coding_standards.pdf Symbian Software Limited, 2008.

[8] Liang Sheng *Java^{TM}Native Interface: Programmer's Guide and Specification* Addison Wesley Professional, 1999.

[9] Rob Gordon *Essential JNI: Java^{TM}Native Interface* Prentice Hall PTR, 1998.

[10] Arvind Gupta and Martin de Jode *Extending the Reach of MIDlets* http://www.arvindgupta.net/pdf/The_Bridge.pdf www.symbian.com, 2005.

[11] C++ component reference in Symbian OS reference http://www.symbian.com/developer/techlib/v9.1docs/doc_source/reference/reference-cpp/index.html

[12] New IPC Mechanisms for Symbian OS http://developer.symbian.com/main/downloads/papers/newipc/new_ipc_mechanisms_for_symbian_os.pdf [10 May 2008].

[13] Hirzel M, Grimm R. Jeannie: Granting Java Native Interface Developers Their Wishes. *ACM SIGPLAN Notices* 2007; **42**(10):19-38.

[14] Bubak M., Kurzyniec D., Luszczek P. Creating Java to native code interfaces with Janet extension. *In Proc. SGI Users's Conference* 2000; 283-294.

[15] JNIWrapper http://www.teamdev.com/jniwrapper/index.jsf [30 May 2008].

[16] Swig http://www.swig.org [30 May 2008].

[17] Dr. Dobb's article of NoodleGlue http://www.ddj.com/java/184401984 [30 May 2008].

FIFTH THIRD EX1062
PART 2 of 3
Page 367 of 379

APPENDIX
HARRISON21

WILEY Online Library

Search

Login / Register

Advertisement

Chapter 4

Introduction to Bluetooth Communication on Mobile Devices

Morten V. Pedersen, Frank H. P. Fitzek

Book Editor(s): Frank H. P. Fitzek, Hassan Charaf

First published: 15 June 2009 | https://doi.org/10.1002/9780470747889.ch4

Series Editor(s): David Hutchison, Serge Fdida, Joe Sventek

PDF CITE TOOLS SHARE



Mobile Peer to Peer (P2P): A Tutorial Guide

Advertisement

Summary

This chapter describes the Bluetooth APIs offered by the Symbian OS. The main focus lies on basic Bluetooth communication issues. Furthermore, the chapter gives several examples covering the most important Bluetooth functionality, such as client discovery and communication issues.

References

developer.symbian.com/main/downloads/papers/bluetooth/Rock_Paper_Scissors_over_Blueto oth.pdf
Google Scholar

Stichbury, J., *'Symbian OS Explained'*, Symbian Press, October 2004.
Google Scholar

Harrison, R. and Shackman, M., *'Symbian OS C++ for Mobile Phones, Vol. 3'*, Symbian Press, June 2007.
Google Scholar

'S60 Platform: Bluetooth API Developer's Guide', available at: sw.nokia.com/id/ac4b09b2-519a-4285-be28-1c19ed9028b9/S60_Platform_Bluetooth_API_Developers_Guide_v2_1_en.pdf [accessed 27 June 2008].
Google Scholar

Bluetooth Core Specifications, www.bluetooth.com [accessed 26 July 2007].
Google Scholar

Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *'Design Patterns: Elements of Reusable Object-Oriented Software'*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1995.
Web of Science® Google Scholar

Issott, A., *'Common Design Patterns for Symbian OS'*, John Wiley & Sons, Ltd, Chichester, UK, October 2008.
Google Scholar

References Related Information

Recommended

Personal Area Networking with Bluetooth
S. Zeadally, A. Kumar
Encyclopedia of RF and Microwave Engineering, Second Edition, [1]

Personal Area Networking with Bluetooth
S. Zeadally, A. Kumar
Encyclopedia of RF and Microwave Engineering, Second Edition, [1]

Bluetooth
From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband, [1]

Advertisement

ABOUT WILEY ONLINE LIBRARY

Privacy Policy

Terms of Use

About Cookies

Manage Cookies

Accessibility

Wiley Research DE&I Statement and Publishing Policies

HELP & SUPPORT

Contact Us

Training and Support

DMCA & Reporting Piracy

Sitemap

OPPORTUNITIES

Subscription Agents

Advertisers & Corporate Partners

CONNECT WITH WILEY

The Wiley Network

Wiley Press Room

WILEY

Copyright © 1999-2026 John Wiley & Sons, Inc or related companies. All rights reserved, including rights for text and data mining and training of artificial intelligence technologies or similar technologies.

APPENDIX
HARRISON22

Symbian Forensics: An Overview

Publisher: IEEE Cite This PDF

Antonio Savoldi ; Paolo Gubian All Authors

4 Cites in Papers | 1 Cites in Patent | 234 Full Text Views

Abstract

Document Sections

1. Introduction
2. Symbian OS Architecture
3. Platform Security
4. Data Collection
5. Experimental Results

Show Full Outline

Authors

Figures

References

Citations

Keywords

Metrics

Abstract:
This paper aims at unveiling the real and complete structure of the Symbian S60 v3 (third edition) platform filesystem, which is considered one of the most secure and armored systems issued by Symbian Ltd. in the last two years. By analyzing the unique features of security, which have been applied at the filesystem level, it will become clear how to develop applications for digital forensic purposes. Moreover, the limitations and main issues related to accessing the observable portion of the filesystem will be pointed out, and possible solutions will be presented.

Published in: 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing

Date of Conference: 15-17 August 2008

DOI: 10.1109/IIH-MSP.2008.239

Date Added to IEEE *Xplore*: 22 August 2008

Publisher: IEEE

Conference Location: Harbin, China

Print ISBN:978-0-7695-3278-3

1. Introduction

Nowadays embedded systems might be considered one of the most pervasive pieces of electronic equipment in our society, able as they are to keep plenty of sensitive and probatory data which are fundamental in a digital forensic scenario. As a matter of fact, a state-of-the-art smartphone, which can be considered as a de facto standard in the embedded device category, might contain phone basic information and SIM-card data, contacts list, caller groups, log records (missed, outgoing and incoming calls), SMS, MMS and E-mail messages (with service data records), calendar events schedule, To-Do items, text notes, photos, video, sound, Java and other files saved in the phone memory or on the flash card, and voice records. Indeed, it is not rhetorical to say that such a platform is getting closer and closer to the traditional desktop/laptop systems, in terms of multimedia capabilities. For instance, a modern Nokia smartphone, which integrates functionalities of a cellular phone plus the PIM part of a PDA, [Sign in to Continue Reading] RAM, 160 Mbytes of internal flash memory, different wireless built-in capabilities, such as Wi-Fi (Wireless Fidelity), Bluetooth, IrDa (Infrared Device Application), GSM (Global System for Mobile Communications), UMTS (Universal Mobile Telecommunications System), HSDPA (High Speed Downlink Packet Access), a built-in high resolution camera, and even a GPS (Global Position System) receiver. Apart from the

Need Full-Text access to IEEE *Xplore* for your organization?
CONTACT IEEE TO SUBSCRIBE >

More Like This

Research on software security and compatibility test for mobile application
Fourth edition of the International Conference on the Innovative Computing Technology (INTECH 2014)
Published: 2014

Software Defined Networking Architecture, Security and Energy Efficiency: A Survey
IEEE Communications Surveys & Tutorials
Published: 2017

Show More

Authors

Figures

References

Export | References & Cited By

Select All

1. M. Breeuwsma. Forensics Imaging of Embedded Systems using JTAG (boundary-scan). In Digital Investigation, volume 3, pages 32-34, 2006.
CrossRef Google Scholar

2. M. Breeuwsma, M. D. Jongh, C. Klaver, R. van der Knijff, and M. Roeloffs. Forensics Data Recovery from Flash Memory. In Small Scale Device Forensics Journal, volume 1, 2007.
Google Scholar

3. I. Campbell. Symbian OS Communication Programming - Second Edition. John Wiley & Sons, Ltd, 2007.
Google Scholar

4. R. Harrison and M. Shackman. Symbian OS C++ for Mobile Phones Volume 3. John Wiley & Sons, Ltd, 2007.
Google Scholar

5. W. Jansen and R. Ayers. An overview and analysis of pda forensic tools. In Digital Investigation, pages 120-132, 2005.
CrossRef Google Scholar

6. M. Jipping. Smart Phone Operating System Concepts with Symbian OS: A Tutorial Guide, John Wiley & Sons, Ltd, 2007.
Google Scholar

7. P. Mokhonoana and M. Olivier. Acquisition of a Symbian Smart Phone's Content with an On-Phone Forensic Tool. In Southern African Telecommunication Networks and Applications Conference 2007 (SATNAC 2007) Proceedings, 2007.
Google Scholar

8. B. Morris. The Symbian OS Architecture Sourcebook: Design and Evolution of a Mobile Phone OS. John Wiley & Sons, Ltd, 2007.
Google Scholar

9. J. Sales. Symbian OS Internals: Real-time Kernel Programming. John Wiley & Sons, Ltd, 2005.
Google Scholar

10. A. Savoldi and P. Gubian. Sim and Usim Filesystem: A Forensic Perspective. In Proceedings of Symposium on Applied Computing (ACM SAC 2007), volume 1, pages 181-187, 2007.
CrossRef Google Scholar

11. A. Savoldi and P. Gubian. Data Hiding and Recovery on Wince Based Handheld Devices. In Fourth Annual IFIP WG 11.9 International Conference on Digital Forensics (IFIP WG 11.9 2008), 2008.
Google Scholar

12. Symbian-Ltd. Symbian OS Library for Application Developers. Available at: www.symbian.com.
Google Scholar

Citations

Keywords

Metrics

IEEE Personal Account

CHANGE USERNAME/PASSWORD

Purchase Details

PAYMENT OPTIONS

VIEW PURCHASED DOCUMENTS

Profile Information

COMMUNICATIONS PREFERENCES

PROFESSION AND EDUCATION

TECHNICAL INTERESTS

Need Help?

US & CANADA: +1 800 678 4333

WORLDWIDE: +1 732 981 0060

CONTACT & SUPPORT

Follow

About IEEE *Xplore* | Contact Us | Help | Accessibility | Terms of Use | Nondiscrimination Policy | IEEE Ethics Reporting | Sitemap | IEEE Privacy Policy
A public charity, IEEE is the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.

© Copyright 2026 IEEE - All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies.

Symbian Forensics: An Overview

Antonio Savoldi and Paolo Gubian
Department of Electronics for Automation
University of Brescia, Via Branze 38, I25121 Brescia, Italy
{antonio.savoldi,paolo.gubian}@ing.unibs.it

Abstract

This paper aims at unveiling the real and complete structure of the Symbian S60 v3 (third edition) platform filesystem, which is considered one of the most secure and armored systems issued by Symbian Ltd. [12] in the last two years. By analyzing the unique features of security, which have been applied at the filesystem level, it will become clear how to develop applications for digital forensic purposes. Moreover, the limitations and main issues related to accessing the observable portion of the filesystem will be pointed out, and possible solutions will be presented.

1. Introduction

Nowadays embedded systems might be considered one of the most pervasive pieces of electronic equipment in our society, able as they are to keep plenty of sensitive and probatory data which are fundamental in a digital forensic scenario. As a matter of fact, a state-of-the-art smartphone, which can be considered as a *de facto* standard in the embedded device category, might contain phone basic information and SIM-card data, contacts list, caller groups, log records (missed, outgoing and incoming calls), SMS, MMS and E-mail messages (with service data records), calendar events schedule, To-Do items, text notes, photos, video, sound, Java and other files saved in the phone memory or on the flash card, and voice records. Indeed, it is not rhetorical to say that such a platform is getting closer and closer to the traditional desktop/laptop systems, in terms of multimedia capabilities. For instance, a modern Nokia smartphone, which integrates functionalities of a cellular phone plus the PIM part of a PDA, might have up to 64 Mbytes of SDRAM, 160 Mbytes of internal flash memory, different wireless built-in capabilities, such as Wi-Fi (Wireless Fidelity), Bluetooth, IrDa (Infrared Device Application), GSM (Global System for Mobile Communications), UMTS (Universal Mobile Telecommunications System), HSDPA (High Speed Downlink Packet Access), a built-in high resolution camera, and even a GPS (Global Position System) receiver. Apart from the increasing rate of diffusion of such devices, we need to ponder about the misuse and abuse of these embedded systems, by increasing the awareness of how it is possible to extract all the digital content from the observable memory of such systems. Presently, one of the major operating systems used in the arena of portable devices is, without doubt, Symbian [8] [9] [6] [4]. One of the first needs for a forensics practitioner is to be able to collect almost all the memory content, RAM and ROM, from such devices, as a preliminary step in a forensic investigation.

The remaining part of the paper has been organized as follows. Firstly, a sound description of Symbian architecture will be provided, by pointing out the basic building blocks of this OS. In addition, the basic elements of the filesystem will be discussed for understanding how to interact with it. Secondly, the new Symbian security platform will be introduced and analyzed thoroughly, giving at the same time a proof of concept about the possibility to gather the full, observable memory content of a Symbian based device. Finally, general guidelines about how to collect probatory elements from the new Symbian platform will be introduced.

2. Symbian OS Architecture

Today's Symbian operating system (OS) might be referred to as EKA2 as well, which means EPOC Kernel Architecture 2. EKA2 is the second iteration of Symbians 32-bit kernel architecture, and this in turn follows 8 and 16-bit kernels designed in the 1980s for Psion personal organizers and PDAs. As can be seen in Figure 1, the architecture is *modular*, where operating system functionalities are provided in separate building blocks, and not in one monolithic unit. Symbian OS is *single user*. Indeed, there is no concept of multiple logins to a Symbian OS smartphone, unlike Windows, Unix or traditional mainframe OSes. In addition, it is *multi-tasking*, being able to switch CPU time between multiple threads, giving the user of the mobile phone the impression that multiple applications are running at the

978-0-7695-3278-3/08 $25.00 © 2008 IEEE
DOI 10.1109/IIH-MSP.2008.239

IEEE computer society

FIFTH THIRD EX1062
PART 2 of 3
Page 375 of 379

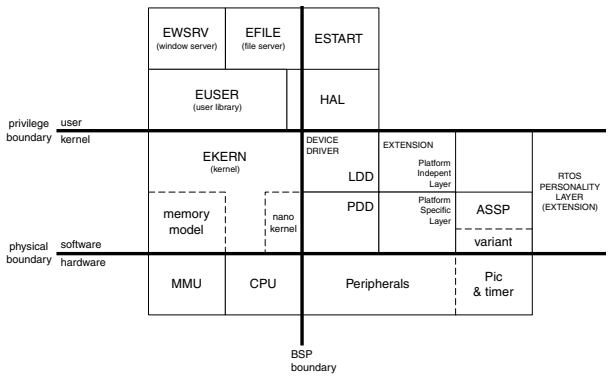


Figure 1. Symbian OS Architecture.

same time. Moreover, it is a *preemptively* multi-tasking OS because it does not rely on one thread to relinquish CPU time to another, but reschedules threads perforce, from a timer tick. It is worth mentioning that Symbian OS is a *priority-based* multi-tasking OS with priority inheritance. The OS allocates CPU time based on a threads priority and minimizes the delays to a high-priority thread when a low-priority thread holds a mutex it needs. Furthermore, Symbian OS is *real-time* based and its services are bounded, that is it completes them in a known amount of time.

The core OS is formed by a microkernel built as a personality layer on top of a real-time (RTOS) nanokernel. This block is responsible for primitives such as fast syncronization, timers, initial interrupts dispatching, and thread scheduling. The Symbian *kernel* might be referred to as the combination of the nanokernel and the microkernel. Interestingly, the nanokernel has been designed to provide just enough functionality to run both the telephone (GSM/GPRS/UMTS) signaling stack, already written for RTOS platforms such as Nucleus and μITRON, and the personal information management (PIM) part on a single processor. This approach provides considerable cost savings over the usual two-processor solution.

Generally speaking, Symbian OS is intended to run on open, small, battery-powered portable computers, that is modern advanced state-of-the-art mobile phones. Besides this, it is designed for 32-bit CPUs, running at low speeds, usually less than 400 MHz, relative to those in desktop or workstations. Today's Symbian OS systems are based on XScale, ARM9 and ARM11-based CPUs, while older devices are powered by the ARM7 family of CPUs.

2.1. Symbian Filesystem

On a Symbian smartphone, the filesystem can be accessed by means of the file server component, also referred to as F32, which manages every file device. It provides services to access the files, directories and drives on those file mapped devices. The file server uses the client/server framework, by receiving and processing file-related requests from multiple clients. Moreover, it is able to deal with different filesystem formats, such as the FAT format used for removable disks, by using components that are plugged into the file server itself. In addition, it supports a maximum of 26 drives, each identified in DOS-like convention by a different drive letter, in the range `A:-Z:`. The main ROM drive, where the firmware resides, is always designated as "`Z:`". This drive holds system executables and data, which are referred to as XIP (eXecutable In Place) because they are directly launched without being loaded into RAM. Besides this, the firmware, or ROM image, is usually programmed into Flash memory, known also as EEPROM, the nonvolatile memory that can be programmed and erased electronically. The `C:` drive is always designated as the main user data drive, which can be mapped onto the same Flash memory chip of the firmware, whereas any removable media device is generally designated as `D:` or `E:`.

It is worth mentioning that every access from a client to file server (F32) takes place via a file server session, by means of RFs server session class, which implements all the basic services to interact with the filesystem. So far, we can obtain information about drives and volumes, act on directories, obtain notification about the state of files and directories, analyze file names, verify the integrity of the filesystem, and finally, manage drives and filesystems.

3. Platform Security

Since the introduction of Symbian OS version 9.1, which can also be referred to as Symbian S60 series third version (S60 v3), Nokia engineers have introduced comprehensive support for new functionality such as Digital Rights Management (DRM), Device Management and Enterprise-grade data handling, which rely on the *Trusted Computing Base* (TCB) concept. At this level of security, the deepest and strongest, every component needs to be certified and formally verified to be secure for having unrestricted access to the device's resources. Only the kernel, the file server, the software installer and its registry are part of the TCB set. Beyond the core of the TCB there is the *Trusted Computing Environment* (TCE), which protects the device resources from misuse, when system components require access to some, but not all, sensitive system resources. The final level of trust is represented by ordinary applications, which need to be authorized to access privileged restricted resources by means of a capability set. This defines an authorization token that can grant access to sensitive APIs such as device drivers or system settings. Indeed, among all APIs there is a subset which can be referred to as *capability-protected* and represents 40% of the Symbian OS APIs. As a conse-

FIFTH THIRD EX1062
PART 2 of 3
Page 376 of 379

quence, an application needs to be authorized, by using a certification process, for using the protected functionality. There are two basic sets of capabilities, basic and extended. The former set protects APIs that access confidential user data, provides data about user's environment and accesses network services. The latter set, the extended group, which requires a phone-manufacture's approval, grants the access to the most reserved area of the phone. This set includes *DRM*, *DiskAccess*, *NetworkControl* and *AllFiles* capabilities, where the last is referred to as the complete access to the filesystem. The following rules will be applied when dealing with capabilities.

- Every process has a set of capabilities and its capabilities never change during its lifetime.
- A process cannot load a DLL that has a smaller set of capabilities than it has itself.
- A DLL cannot statically link to a DLL that has a smaller set of capabilities than it has itself.
- The access rules of a file are entirely determined by its directory path, regardless of the drive.

The latter rule explains how to control the file access by means of the various processes running on the phone. This solution is different from a traditional access control list, defined per process, being based on a fixed access control policy which is equally valid for all the processes. This concept might be referred to as *data caging* and there are four different sets of access rules, which are represented by four different directory hierarchies under the root of the filesystem, according to the following scheme.

- **sys** Only TCB granted processes can read and write files under this directory, which contains system and installed binaries, vital for the integrity of the platform.
- **resource** All processes can read files in this directory, but only TCB processes can write to them. In this directory we might find fonts and help files that are not expected to change once installed.
- **private** Every program has its own private subdirectory, with the name <application SID>, where SID is a secure identifier which uniquely identifies it. Only the granted processes and the backup server can read and write files in a process's private directory. Other processes may neither read nor write.
- **Other root files and directories** The remaining portion of the filesystem has no restrictions in reading or writing, being considered public and thus accessible space.

So far, if an application needs to have the complete access (e.g. read) to the phone filesystem, it has to be authorized by means of the Symbian signing procedure with *AllFiles* capabilities, which requires a special certificate released by TC Trust Center. Three steps are required to sign an application. Initially, the installation file generator, `makesis.exe`, creates the installation files (extension `.sis`) from information specified in the package file (extension `.pkg`). After that, if the application is for the international market, it will be signed with an ACS Publisher ID, by means of the Symbian Signed service. Conversely, it will be signed with a user-generated certificate, which might be created with `makekeys.exe`. Finally, the Installation File Signer (`signsis.exe`), digitally signs the installation files with the proper certificate, by generating, as a result, a `.sisx` file.

4. Data Collection

After the exhaustive preamble of previous paragraphs it is worth pointing out what the main issues are when dealing with the implementation of an open source application, for forensic purposes, which aims at collecting the entire set of digital objects present on a Symbian mobile phone filesystem. It should be evident that with the new Symbian security platform the collection of the entire, observable, memory content is restricted by means of data caging, at least for `C:` user data disk, where all probatory data of interest are located. This implies having the phone-manufacturer approval to be able to gather the full and complete filesystem content. Interestingly, the filesystem restriction policy is fully contained in the file known as `SWIPOLICY.INI` [12], located in the folder `z:\system\data\`. The original policy file related to a Nokia E65 Symbian based smartphone is illustrated as follows.

```
AllowUnsigned = false
MandatePolicies = false
MandateCodeSigningExtension = false
Oid = 1.2.3.4.5.6
Oid = 2.3.4.5.6.7
DRMEnabled = true
DRMIntent = 3
OcspMandatory = false
OcspEnabled = true
AllowGrantUserCapabilities = true
AllowOrphanedOverwrite = true
UserCapabilities = NetworkServices LocalServices
ReadUserData WriteUserData UserEnvironment
AllowPackagePropagate = true
SISCompatibleIfNoTargetDevices = false
RunWaitTimeoutSeconds = 600
AllowRunOnInstallUninstall = false
DeletePreinstalledFilesOnUninstall = true
```

It is interesting to observe that the capability set defined in the previous file is limited, and it restricts the interaction with the filesystem of the mobile platform. According to

FIFTH THIRD EX1062
PART 2 of 3
Page 377 of 379

the standard documentation, the various parameters can appear in any order. Moreover, `UserCapabilities` set might be changed, by adding the required capabilities such as those illustrated in the following modified version of policy file.

```
AllowUnsigned = true
MandatePolicies = false
MandateCodeSigningExtension = false
Oid = 1.2.3.4.5.6
Oid = 2.3.4.5.6.7
OcspMandatory = false
OcspEnabled = true
AllowGrantUserCapabilities = true
UserCapabilities = AllFiles DiskAdmin NetworkServices
LocalServices ReadUserData WriteUserData
UserEnvironment MultiMediaDD NetworkControl CommDD
ReadDeviceData WriteDeviceData
SISCompatibleIfNoTargetDevices = false
AllowRunOnInstallUninstall = true
AllowPackagePropagate = true
DeletePreinstalledFilesOnUninstall = true
```

The illustrated policy file can be written directly in the original firmware of the phone and, subsequently, uploaded by means of reflashing. As a result, the complete `C:` disk content might be collected, with standard self-signed APIs, and thus analyzed, to extract the full set of probatory data which might be usually found on a mobile platform. This is the usual approach for other mobile platforms such as Windows CE [11] or for traditional mainframe systems, where the primary image, the one which contains the entire set of evidence, can be obtained without any restrictions. Unfortunately, such a scenario is far from the reality, and we need to evaluate others ways to access the digital data content of the smartphone.

General guidelines for PDA and smartphone forensics have been released by [5], where the authors pointed out that investigators cannot change the digital data content of the device being analyzed. Moreover, an audit trail of the investigation process, suitable for independent third-party review, must be created and preserved, accurately documenting each investigative step. Finally, the person in charge of the investigation has the overall responsibility for ensuring that the above-mentioned procedures are followed and are in compliance with the governing laws. Another important issue to be aware of is the general set of requirements that a forensically sound method should have when used on a mobile device. The method should minimize changes on the device, be able to retrieve the full set of data, and finally minimize user interaction with the device itself.

Ideally, the full memory content of a generic embedded device should be collected, to preserve the full inner state and obtain a forensically sound acquisition. As a matter of fact, with Symbian S60 platform this is not possible. Nevertheless, as detailed by [7], there is a considerable number of other approaches to capture data from the mobile platform, which are as follows.

4.1. Manual examination

This approach requires a manual inspection of the device being investigated, by means of an external recording device (e.g. photocamera) which is able to take snapshots of visual evidence. The method is error-prone because of the direct interaction between the investigator and the device, which can cause an easy alteration of the content. It should be used only when there are no other ways to extract digital data, and a sound and complete audit trail of the investigative procedure must be provided for third-party examination.

4.2. Connectivity services

This is a software approach based on command-response protocols, such as AT Command Set, SyncML, OBEX, Nokia FBUS proprietary protocol or applications based on Nokia PC Suite Connectivity [3]. All mentioned protocols are unable to have the full and complete access to the memory areas where probatory data are located, the `C:` and external drive disks. Even so, such protocols might be used for collecting digital objects at the logical level, by ensuring that no command will be issued to modify the content of the phone.

4.3. Connection agent

This method, again software based, uses a specific program, the so-called connection agent, which is installed on the target device. It is capable of establishing a connection and exchanging data with an externally connected host computer. The main issue with this approach is that it requires a piece of software to be loaded on the device, thus modifying the target device. This kind of approach has been successfully used by the authors with Windows CE platform [11].

4.4. Hardware approach

This method permits to obtain a forensically sound copy of the entire flash memory content, making it possible to access deleted or partially overwritten entries of the filesystem. This might be obtained by means of the *JTAG* test access port of an embedded device. A JTAG port is used for debugging purposes, but can also be used to access the flash memory [1]. Despite the difficulties related to this process, which depends on the possibility to easily find the hardware interface, it can be guaranteed that no data is written during the collection phase. The other possible approach to collect data from embedded devices' memory is by *de-soldering* the flash chips from the board and reading the content with a memory chip programmer or reader. Although this method is the most invasive for the equipment, it guarantees that

FIFTH THIRD EX1062
PART 2 of 3
Page 378 of 379

data can be recovered even from broken or damaged embedded systems.

4.5. Service provider

In the GSM/UMTS network environment, a great deal of information might be recovered from the service provider [7]. The set of information which can be successfully collected with this method is related to the SIM/USIM data set, such as SMS, MMS, list of last called numbers, and the location of the subscriber [10]. Clearly, information such as photos, videos, phone book, web browser logs, audio recordings, or user's notes cannot be gathered in such a way.

5. Experimental Results

So far, Nokia S60 v3 (third edition) mobile phone platform does not permit to acquire the complete flash memory content at the logical level, by means of software techniques, for the limitation which has been introduced with the new security model, which relies on data caging. With the previous Nokia phone generations, for instance the S40 series, the logical acquisition of the device content was possible, and almost trivial, by means of Symbian OS APIs, which were able to copy the entire filesystem content on an external memory device (e.g. SD card) [2]. This approach might be used with the newer Symbian phones (e.g. Nokia S60 devices) only when the OS has been provided with the "AllFile" capabilities, as it has already been shown with regard to the E65 Nokia phone example. Is such a case, as it has been demonstrated, the whole logical content might be acquired. For instance, by interpreting the raw data objects of the collected filesystem, we can carve out all the information related to SMS and MMS. Every object of this type is saved in a file with a name that follows the pattern `00100XXX`, with `XXX` a hexadecimal number. All the files are saved according to the following path.

```
C\Private\1000484b\Mail2\00001001_S\[0..F]
..\Mail2\00001001_S\0001000c0
..\Mail2\00001001_S\0001001a0
..\Mail2\00001001_S\0001001c0
..\Mail2\00001001_S\000100020
..\Mail2\00001001_S\000100040
..\Mail2\00001001_S\000100080
..\Mail2\00001001_S\000100280
```

6. Conclusions

To summarize, Digital forensic methodologies for the new Symbian S60 v3 OS based phones are still in their infancy stage, the security paradigm being new and very restrictive towards filesystem access. Software approaches for imaging tool might only be used for acquiring specific items. For instance, it is certainly possible to extract the entire set of probatory data, such as SMS, MMS, pictures, video clip and phone book, by using Symbian application APIs. From an integrity perspective, the hardware approach seems the only one which should be able to give a bit-by-bit image of the flash memory content thus preserving the content of the investigated phone.

References

[1] M. Breeuwsma. Forensics Imaging of Embedded Systems using JTAG (boundary-scan). In *Digital Investigation*, volume 3, pages 32–34, 2006.

[2] M. Breeuwsma, M. D. Jongh, C. Klaver, R. van der Knijff, and M. Roeloffs. Forensics Data Recovery from Flash Memory. In *Small Scale Device Forensics Journal*, volume 1, 2007.

[3] I. Campbell. *Symbian OS Communication Programming - Second Edition*. John Wiley & Sons, Ltd, 2007.

[4] R. Harrison and M. Shackman. *Symbian OS C++ for Mobile Phones Volume 3*. John Wiley & Sons, Ltd, 2007.

[5] W. Jansen and R. Ayers. An overview and analysis of pda forensic tools. In *Digital Investigation*, pages 120–132, 2005.

[6] M. Jipping. *Smart Phone Operating System Concepts with Symbian OS: A Tutorial Guide*. John Wiley & Sons, Ltd, 2007.

[7] P. Mokhonoana and M. Olivier. Acquisition of a Symbian Smart Phone's Content with an On-Phone Forensic Tool. In *Southern African Telecommunication Networks and Applications Conference 2007 (SATNAC 2007) Proceedings*, 2007.

[8] B. Morris. *The Symbian OS Architecture Sourcebook: Design and Evolution of a Mobile Phone OS*. John Wiley & Sons, Ltd, 2007.

[9] J. Sales. *Symbian OS Internals: Real-time Kernel Programming*. John Wiley & Sons, Ltd, 2005.

[10] A. Savoldi and P. Gubian. Sim and Usim Filesystem: A Forensic Perspective. In *Proceedings of Symposium on Applied Computing (ACM SAC 2007)*, volume 1, pages 181–187, 2007.

[11] A. Savoldi and P. Gubian. Data Hiding and Recovery on Wince Based Handheld Devices. In *Fourth Annual IFIP WG 11.9 International Conference on Digital Forensics (IFIP WG 11.9 2008)*, 2008.

[12] Symbian-Ltd. *Symbian OS Library for Application Developers*. Available at: www.symbian.com.

FIFTH THIRD EX1062
PART 2 of 3
Page 379 of 379