Multiplayer Games in Mobile Terminals

Mattias Hedlund
Robert Maxe
Abstract

Today’s mobile terminals have many similarities with personal computers, for example, the possibility to add new applications. The Java programming language is known for its mobility between platforms and it is available for mobile terminals. TeliaSonera have a mobile network that can be used for data services. However, it has different characteristics than the high-speed low-latency Internet.

The annual sales of computer games in the world are estimated to 30 billion dollars and mobile terminals can introduce computer games to new situations. However, multiplayer games have not yet been a success for mobile terminals. The purpose is therefore to do a literature study in four areas of importance for the development and commercialisation of mobile multiplayer games and to develop a demonstration version of a multiplayer game for a mobile terminal.

A multiplayer game is developed and the steps in the development process are described from the initial game idea to the network latency measurements of the finished game. The result of the measurements shows that mobile networks are still evolving but latency tolerant multiplayer games are possible today.
Acknowledgements

Firstly, we would like to thank TeliaSonera Sverige AB for allowing us to work with this master’s thesis and for providing an office in Sundsvall, Sweden. We will remember this period as instructive and inspiring.

Secondly, we would like to thank the following people for their help during our work:

- Markus Marklund and Peter Sirviö, our supervisors at TeliaSonera, for being a support in any matter and for their help to find the necessary equipment.
- Fredrik Bengtsson, our examiner at Luleå University of Technology, for his constructive comments on chapter drafts.
- Christer Lundqvist and Robert Tjernström, both at TeliaSonera, for their willingness to share their knowledge about the GPRS-network.
- Gunnar Hägglund, at TeliaSonera, for measuring the network round-trip time.
- Lars Liljestam, at TeliaSonera, for a dialogue concerning the structure of the report.

Finally, we would like to thank all our colleagues at TeliaSonera for the cheerful atmosphere and the interesting discussions during the coffee breaks.


| Mattias Hedlund | Robert Maxe |
Table of Contents

1 INTRODUCTION ................................................................................................................ 14
  1.1 BACKGROUND ............................................................................................................... 14
  1.2 PURPOSE ........................................................................................................................ 15
  1.3 OBJECTIVES ................................................................................................................... 15
  1.4 DELIMITATIONS ............................................................................................................. 15

2 MULTIPLAYER GAMES .................................................................................................. 16
  2.1 USED WHILE WAITING .................................................................................................. 16
  2.2 SOCIAL .......................................................................................................................... 17
  2.3 CHALLENGING ............................................................................................................... 17
  2.4 FUN ................................................................................................................................ 18
  2.5 CURRENT GAMES .......................................................................................................... 19

3 MOBILE TERMINALS ...................................................................................................... 20
  3.1 OPERATING SYSTEMS .................................................................................................... 20
    3.1.1 Symbian .................................................................................................................... 20
    3.1.2 Windows Mobile ....................................................................................................... 21
  3.2 PROGRAMMING LANGUAGES ......................................................................................... 21
    3.2.1 Java 2 Micro Edition ............................................................................................... 21
      3.2.1.1 Configurations ............................................................................................................. 21
      3.2.1.2 Profiles ........................................................................................................................ 22
      3.2.1.3 Optional Packages ....................................................................................................... 24
      3.2.1.4 Application Programming Interface ............................................................................ 24
    3.2.2 C++ .......................................................................................................................... 26
  3.3 HARDWARE CAPABILITIES ............................................................................................ 27

4 NETWORK CHARACTERISTICS ................................................................................... 30
  4.1 CONNECTION M odes .................................................................................................... 30
    4.1.1 Circuit Switched Data .............................................................................................. 30
    4.1.2 Packet Switched Data .............................................................................................. 30
  4.2 GSM .............................................................................................................................. 31
  4.3 GPRS ............................................................................................................................. 32
    4.3.1 Time Slots .................................................................................................................. 33
    4.3.2 Multi-slot Classes ..................................................................................................... 34
    4.3.3 Channel Coding ......................................................................................................... 35
    4.3.4 Bandwidth ................................................................................................................ 35
    4.3.5 Latency ...................................................................................................................... 36
5 MULTIPLAYER BUSINESS ................................................................. 39
5.1 PRESENT STATE ................................................................. 39
5.2 NECESSARY INFRASTRUCTURE ........................................... 40
5.3 TERMINALS ............................................................................ 40
  5.3.1 Requirements for Java Games ............................................. 40
  5.3.2 Availability ........................................................................ 41
  5.3.3 Easiness of Game Installation ............................................ 42
5.4 CUSTOMER INTEREST ............................................................ 43
5.5 BUSINESS PARTIES ................................................................. 43
5.6 CHARGING METHODS ............................................................ 45
5.7 REVENUE FROM GAMES ....................................................... 45
5.8 DATA TRANSFER PER GAME SESSION ................................... 46
5.9 GAME DEVELOPMENT PROJECTS ............................................ 46

6 DEVELOPMENT OF A MULTIPLAYER GAME ................................ 47
6.1 GAME IDEAS .......................................................................... 47
  6.1.1 The Right Way (implemented) ............................................. 47
  6.1.2 Guess Who ........................................................................ 48
  6.1.3 Four in a Row .................................................................... 49
6.2 CHOICE OF PROGRAMMING LANGUAGE ................................ 49
6.3 THE TARGETED TERMINAL ................................................... 50
6.4 USING THE GAME WITH OTHER TERMINALS ......................... 51
6.5 DEVELOPMENT TOOLS .......................................................... 53
6.6 INCLUDED MULTIPLAYER FEATURES ..................................... 53
6.7 MOTIVATIONS OF GAME FEATURES ....................................... 54
  6.7.1 Easy for Beginners ............................................................ 54
  6.7.2 Short Game Sessions ........................................................ 54
  6.7.3 Customized Players .......................................................... 55
  6.7.4 Background Light ............................................................. 55
  6.7.5 No Sounds ........................................................................ 55
  6.7.6 Feedback when a Button is pressed .................................... 55
  6.7.7 Standard Component Menus .............................................. 56
  6.7.8 Use of a Game Server ....................................................... 56
  6.7.9 Placement of Game States ................................................. 56
  6.7.10 HTTP Connection Type .................................................. 57
  6.7.11 The Server is polled for Updates ..................................... 57
  6.7.12 A Wait Screen during Network Access ............................. 58
6.8 LATENCY MEASUREMENTS .................................................... 59
  6.8.1 Round-Trip Time ............................................................. 59
  6.8.2 In-Game Request Time ..................................................... 61
6.9 PROBLEMS DURING DEVELOPMENT ...................................... 68
7 SYSTEM DESCRIPTION................................................................................................... 71

7.1 CLIENT ........................................................................................................................... 71
7.1.1 Menu System ............................................................................................................ 71
7.1.2 Class Diagram ......................................................................................................... 71
7.1.3 Game Walk-Through............................................................................................... 75
7.1.4 Labyrinth Construction............................................................................................ 76

7.2 SERVER .......................................................................................................................... 76
7.2.1 State Chart ............................................................................................................... 76
7.2.2 Class Diagram ......................................................................................................... 78

7.3 CLIENT-SERVER COMMUNICATION ........................................................................ 80

7.4 LIMITATIONS OF THE DEMONSTRATION VERSION ................................................. 83

7.5 IDEAS THAT CAN IMPROVE THE GAME .................................................................. 84

8 CONCLUSIONS................................................................................................................... 86

9 REFERENCES ..................................................................................................................... 88

APPENDIX A: INSTALLATION INSTRUCTIONS................................................................. 94

A.1 GAME CLIENT.............................................................................................................. 94
A.2 GAME SERVER............................................................................................................. 95

APPENDIX B: DATA TABLES.............................................................................................. 97
List of Figures

Figure 1: Play a game while waiting on a bus................................................................. 16
Figure 2: Buy a new game in the middle of the night..................................................... 17
Figure 3: A typical keypad on a mobile phone............................................................... 28
Figure 4: Distribution of the components in a GSM- and GPRS-network..................... 32
Figure 5: A user connected with CSD reserves one time slot during the entire call...... 33
Figure 6: A user connected with HSCSD can reserve more than one time slot............ 33
Figure 7: Two GPRS-users share the same time slots................................................... 34
Figure 8: The new radio access network of UMTS is connected to GSM and GPRS core networks.............................................................. 38
Figure 9: Current distribution of game fees in Europe.................................................... 39
Figure 10: Distribution of global handset shipments in quarter two 2003 [percentage].................................................................................................................. 42
Figure 11: The traffic fee from a multiplayer game can be shared between a network operator and a game provider................................................................. 44
Figure 12: Providing distribution services can increase the revenue for a network operator.............................................................................................................. 44
Figure 13: The target groups for games:......................................................................... 46
Figure 14: Three players in a labyrinth.......................................................................... 48
Figure 15: Labyrinth contents: start, exit, light, delay and blackout............................... 48
Figure 16: A Nokia 7650 terminal................................................................................... 51
Figure 17: Different utilization of the screen size............................................................. 52
Figure 18: A network topology for a two player competition......................................... 60
Figure 19: Distribution of minimum-, maximum- and average latency for UMTS........ 63
Figure 20: Distribution of minimum-, maximum- and average latency for GPRS using WAP........................................................... 64
Figure 21: Distribution of minimum latency for GPRS using WAP................................. 64
Figure 22: Distribution of maximum latency for GPRS using WAP................................ 65
Figure 23: Distribution of average latency for GPRS using WAP.................................. 65
Figure 24: Distribution of minimum-, maximum- and average latency for GPRS using TCP............................................................................................................. 66
Figure 25: Distribution of minimum latency for GPRS using TCP................................. 66
Figure 26: Distribution of maximum latency for GPRS using TCP................................ 67
Figure 27: Distribution of average latency for GPRS using TCP.................................. 67
Figure 28: The visibility of labyrinth walls was enhanced in response to feedback from end-user tests................................................................. 69
Figure 29: Client menu system........................................................................................ 72
Figure 30: Client class relations...................................................................................... 73
Figure 31: Client class contents...................................................................................... 74
FIGURE 32: THE SERVLET FLOW DIAGRAM SHOWING THE ONLY STATE AND ALL THE ACTIONS...... 77
FIGURE 33: SERVER CLASS RELATIONS................................................................. 78
FIGURE 34: SERVER CLASS CONTENTS............................................................... 79
FIGURE 35: A PACKET SENT FROM THE CLIENT WITH A REQUEST TO JOIN A GAME. .................. 80
FIGURE 36: A PACKET SENT FROM THE SERVER IN REPLY TO A REQUEST TO JOIN A GAME. ........... 80
FIGURE 37: PACKET SPECIFICATIONS........................................................................ 81
List of Tables

Table 1: The first 12 multi-slot classes and their time slots ............................................ 34
Table 2: Data rates for Coding Scheme 1 to 4 ................................................................. 35
Table 3: Data from the GPRS latency test using TCP ....................................................... 97
Table 4: Data from the GPRS latency test using WAP .................................................... 98
Table 5: Data from the UMTS latency test ...................................................................... 98
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1G</td>
<td>First generation</td>
<td>Refers to the NMT-network and its equals.</td>
</tr>
<tr>
<td>2G</td>
<td>Second generation</td>
<td>Refers to the GSM-network and its equals.</td>
</tr>
<tr>
<td>2.5G</td>
<td>Second and a half generation</td>
<td>Refers to the GPRS-network and its equals.</td>
</tr>
<tr>
<td>3G</td>
<td>Third generation</td>
<td>Refers to the UMTS-network and its equals.</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
<td>Used to control computer opponents in a game.</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
<td>An interface that makes it easier for programmers to develop software.</td>
</tr>
<tr>
<td>BSC</td>
<td>Base Station Controller</td>
<td>A node in GSM/GPRS access network that controls several BTSs and routes information to a MSC/SGSN.</td>
</tr>
<tr>
<td>BSS</td>
<td>Base Station Subsystem</td>
<td>The GSM/GPRS access network, including BTSs and BSC.</td>
</tr>
<tr>
<td>BTS</td>
<td>Base Transceiver Station</td>
<td>Handles the radio signals to and from the mobile stations.</td>
</tr>
<tr>
<td>CDC</td>
<td>Connected Device Configuration</td>
<td>A configuration for wireless terminals with at least a few megabytes of memory.</td>
</tr>
<tr>
<td>CGI</td>
<td>Common Gateway Interface</td>
<td>A technique for making documents on the Internet interactive.</td>
</tr>
<tr>
<td>CLDC</td>
<td>Connected Limited Device Configuration</td>
<td>A configuration for wireless resource-constrained terminals like mobile phones and pagers.</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
<td>A unit that makes computations in a computer.</td>
</tr>
<tr>
<td>CS</td>
<td>Coding Scheme</td>
<td>Used to ensure higher reliability.</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>CSD</td>
<td>Circuit Switched Data. Create a connection by reserving network resources between two terminals.</td>
<td></td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System. The DNS is used in the Internet to translate domain names to IP-addresses.</td>
<td></td>
</tr>
<tr>
<td>FDMA</td>
<td>Frequency Division Multiple Access. A radio access technology. The available frequency spectra for are divided into channels.</td>
<td></td>
</tr>
<tr>
<td>GCF</td>
<td>Generic Connection Framework. Provides an abstraction of seven different types of connections.</td>
<td></td>
</tr>
<tr>
<td>GGSN</td>
<td>Gateway GSN. A node in GPRS responsible for connections with other packet-based networks, for example, the Internet.</td>
<td></td>
</tr>
<tr>
<td>GMSC</td>
<td>Gateway MSC. A node in GSM responsible for connections to and from fixed networks, for example, the public switched telephone network.</td>
<td></td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service. A packet switched network built upon GSM with better resource utilization and higher bandwidth. Also known as 2.5G.</td>
<td></td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System. A system with navigation satellites used for positioning services.</td>
<td></td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile communications. A digital and circuit switched network for mobile terminals. Also known as 2G.</td>
<td></td>
</tr>
<tr>
<td>GSN</td>
<td>GPRS Support Nodes. Forms the IP-based backbone network in GPRS.</td>
<td></td>
</tr>
<tr>
<td>HSCSD</td>
<td>High Speed CSD. A technique for sending and receiving CSD at high speeds.</td>
<td></td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language. A standardized language for web pages.</td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol. A transfer protocol mainly used to transfer web pages. The duration for transferring a HTTP-request from the game client to the game server, for the server to process the request and for the HTTP-response to be transferred back to the game client using some transport layer</td>
<td></td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
<td></td>
</tr>
<tr>
<td>ITU-T</td>
<td>International Telecommunication Union - Telecommunication</td>
<td></td>
</tr>
<tr>
<td>J2EE</td>
<td>Java 2 Enterprise Edition</td>
<td></td>
</tr>
<tr>
<td>J2ME</td>
<td>Java 2 Micro Edition</td>
<td></td>
</tr>
<tr>
<td>J2SE</td>
<td>Java 2 Standard Edition</td>
<td></td>
</tr>
<tr>
<td>JAD</td>
<td>Java Application Descriptor</td>
<td></td>
</tr>
<tr>
<td>JAR</td>
<td>Java Archive</td>
<td></td>
</tr>
<tr>
<td>JSP</td>
<td>Java Server Pages</td>
<td></td>
</tr>
<tr>
<td>KVM</td>
<td>Kilobyte Virtual Machine</td>
<td></td>
</tr>
<tr>
<td>MIDP</td>
<td>Mobile Information Device Profile</td>
<td></td>
</tr>
<tr>
<td>MIME</td>
<td>Multipurpose Internet Mail Extensions</td>
<td></td>
</tr>
<tr>
<td>MMS</td>
<td>Multimedia Messaging Service</td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>Mobile Station</td>
<td></td>
</tr>
<tr>
<td>MSC</td>
<td>Mobile Switching Centre</td>
<td></td>
</tr>
<tr>
<td>NMT</td>
<td>Nordic Mobile Telephony</td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
<td></td>
</tr>
</tbody>
</table>

- **IP**: A protocol allowing various networks and nodes to communicate with each other.
- **ITU-T**: Study groups ensuring an efficient and on-time production of high quality communication standards.
- **J2EE**: Edition of the Java 2 platform targeted at server applications.
- **J2ME**: Edition of the Java 2 platform targeted at consumer electronics and embedded devices.
- **J2SE**: Edition of the Java 2 platform targeted at desktop computers.
- **JAD**: A JAD-file enables the installer to verify that a Midlet suites the phone before downloading the JAR-file of a Midlet suite.
- **JAR**: A Java archive contains the class- and resource files associated with a Midlet.
- **JSP**: A technology to combine static HTML and dynamic HTML in web pages.
- **KVM**: A virtual machine (VM) running on just tens of kilobytes of memory.
- **MIDP**: A profile defining the Java application environment for limited hardware devices, for example, mobile phones.
- **MIME**: Informs the client what type of content is to be received as result of a HTTP-request.
- **MMS**: An extension of SMS to send messages with pictures and sounds.
- **MS**: A terminal in a mobile network.
- **MSC**: A node in GSM that is responsible for routing calls, tracking mobile users and security functions.
- **NMT**: The first analogue mobile network. Also known as 1G.
- **OS**: A collection of programs on a computer to handle resources and
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTA</td>
<td>Over-the-Air Wireless data transfer, in a mobile network.</td>
</tr>
<tr>
<td>PCU</td>
<td>Packet Control Unit A node in the GSM/GPRS access network responsible for the GPRS radio resource management.</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant A small hand-held computer.</td>
</tr>
<tr>
<td>PLMN</td>
<td>Public Land Mobile Network GSM is an example of a PLMN.</td>
</tr>
<tr>
<td>PSD</td>
<td>Packet Switched Data Large chunks of data to be sent are divided into standardized packets with header fields and payload.</td>
</tr>
<tr>
<td>PSTN</td>
<td>Public Switched Telephone Network A regular telephone network (non-mobile).</td>
</tr>
<tr>
<td>RAN</td>
<td>Radio Access Network The UMTS access network, including RBSs and RNC.</td>
</tr>
<tr>
<td>RBS</td>
<td>Radio Base Station Handles the radio signals to and from the mobile stations.</td>
</tr>
<tr>
<td>RNC</td>
<td>Radio Network Controller A node in UMTS access network that controls several RBSs and routes information to the GSM/GPRS network.</td>
</tr>
<tr>
<td>RTT</td>
<td>Round-Trip Time A measure of the current delay on a network. The time for a request to reach a remote host and back again.</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit A set of tools used to produce software, for example, a text editor, a compiler and a debugger.</td>
</tr>
<tr>
<td>SGSN</td>
<td>Serving GSN A node in GPRS that is responsible for routing calls, tracking mobile users and security functions.</td>
</tr>
<tr>
<td>SIP</td>
<td>Session Initiation Protocol A protocol that initiates sessions for other protocols between hosts.</td>
</tr>
<tr>
<td>SMS</td>
<td>Short Message Service A service to send short text messages, for example, in the GSM-network.</td>
</tr>
<tr>
<td>SS7</td>
<td>Signalling System no. 7 A system for signalling within telephone networks.</td>
</tr>
<tr>
<td>TDMA</td>
<td>Time Division Multiple Access A radio access technology. The users transmit in turns. Each one use a time slot.</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications A mobile network that enables video-calls and music streaming</td>
</tr>
<tr>
<td>Short Form</td>
<td>Full Form</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>System</td>
<td>System with high data rates. Also known as 3G.</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
<tr>
<td>WAV</td>
<td>Waveform</td>
</tr>
<tr>
<td>WCDMA</td>
<td>Wideband Code Division Multiple Access</td>
</tr>
</tbody>
</table>
1 Introduction

This chapter gives a background to multiplayer games in mobile terminals and states the purpose of the report. To help the authors achieve the purpose, a number of objectives and delimitations are listed.

1.1 Background

The annual sales of computer games are 30 billion dollars in the world and four billion dollars in Western Europe according to analysts (Dagens industri 2003). The large sales make it desirable to capture a part or even a fraction of the market for games. An example of a popular game is Counter-Strike (CS). This multiplayer game has 2.5 million players worldwide with more than 100,000 playing at any given moment (Microsoft 2004).

Today’s mobile terminals have many similarities with personal computers (PCs). Examples of common features are a colour display and sound playback. In addition, the processing power of a mobile terminal was state-of-the-art for PCs less than ten years ago. The Java programming language is known for its mobility between platforms, which makes it possible to run the same version of an application on different hardware. Java is available for mobile terminals with the Java 2 Micro Edition (J2ME), which makes it possible for a user to add new applications to a mobile terminal, for example, a game.

TeliaSonera have a mobile network that supports voice calls and data services. This network can be used to transfer in-game data and thereby enable multiplayer games for mobile terminals. However, a mobile network has different characteristics than the Internet, which it is connected to. As a result, some multiplayer game types developed for PCs may not be suited for mobile terminals. On the other hand, mobile terminals can be used anywhere, and thereby introduce games in new situations. Nevertheless, multiplayer games for mobile terminals have not yet been a success.
1.2 Purpose

A theoretical and an empirical part have been identified. The purpose of the theoretical part is to do a literature study in four areas of importance for the development and commercialisation of mobile multiplayer games. The four areas are:

- the programming languages
- the terminals
- the mobile networks
- the business

With this information as a foundation, the empirical part has the purpose to develop a demonstration version of a multiplayer game for a mobile terminal. The experiences gained will result in recommendations on what role TeliaSonera is to play.

1.3 Objectives

1. Examine which multiplayer games are available today.
2. Examine which technologies are needed to enable multiplayer games.
3. Examine the game development capabilities in one of the available programming languages.
4. Examine the characteristics of a mobile network.
5. Evaluate which types of multiplayer games are suited for mobile terminals in mobile networks.
6. Evaluate the possibilities of multiplayer games from a business point of view.
7. Develop a multiplayer game to demonstrate the capabilities of a mobile network and a mobile terminal.
8. Discuss the future of multiplayer games in mobile terminals.

1.4 Delimitations

1. In objectives 3 and 7, the programming language focus will be on Java.
2. The communication between game clients will utilize and focus on mobile networks.
3. The targeted terminal will be a mobile phone named Nokia 7650.
4. No end-users will be involved in the evaluation in objective 5.
Chapter 2: Multiplayer Games

2 Multiplayer Games

This chapter describes a new situation where mobile games can be played, where stationary equipment cannot be used. It also motivates games in general and multiplayer games in particular. A brief orientation of the current multiplayer games is also included.

2.1 Used WhileWaiting

A mobile terminal is often one of the things that are brought when leaving home, together with keys and a wallet. This availability means that it is possible to play in a variety of new situations compared to stationary game equipment, for example, when waiting for transportation, which is illustrated in Figure 1, or waiting for a person to come by. In these situations the game sessions tend to be short, since the awaited finally arrives. Therefore, games should be easy to start, interrupt and to save. However, in a single-player game, only the game state has to be saved to allow a player to resume at a later time. In a multiplayer game, the moment with the opponents is lost. However, if the players can resume on their own, the results can be compared afterwards. Solutions to this problem are discussed in chapter 6.7.2.

A factor that enables spontaneous game play is the possibility to download new games over-the-air (OTA). The main advantage with this technique is that the distribution of games is not tied to a specific time or place. On the contrary, a game that is distributed on a memory card has to be bought in a game shop. This shop may not be in the vicinity and it may also be closed, for example, late at night, which is illustrated in Figure 2. Thus, the game session has to be planned in advance. Nevertheless, the Nokia N-Gage terminal uses memory cards. The
reason stated is that games with intense graphics and sounds are too large to be downloaded OTA (Nokia 2003d). Distribution on memory cards also limits game development to a number of approved suppliers, which help to ensure the quality of games (Nokia 2003e).

2.2 Social

People that share the interest to play games may develop an affinity to other players. It gives them something to talk about. Accordingly, if a chat facility is provided, players can discuss what happened in a game session. It can also be interesting to talk with people who did not participate in a specific session. Inexperienced players are able to receive help from other players and a leader board is always something to talk about (Nokia 2003a).

When a person is alone for a moment, it feels comforting to find other players online, which can be in the same situation. Another social factor is the possibility to team up with another player and co-operate to reach a common goal in a game.

Figure 2: Buy a new game in the middle of the night.

People like to distinguish themselves form others (Nokia 2003a). Therefore, it is important to allow people to customise their appearance in a game. The customisation can include a choice of character and to let the player provide its name (Nokia 2003a).

2.3 Challenging

There are mainly 3 types of challenges in games: physical, mental and the opposition (Nokia 2003a). Often, more than one type of challenge is used in a game. In that way games are more varied and a successful player are required to have different kinds of skills.

Physical challenges means that the player must give input to the game at the right moments. An example can be to avoid an obstacle at a racetrack. Mental challenges are a little different. They occur when the player has to solve some a problem or puzzle. Small puzzles can be included in many types of games; examples are resource trade-offs or tricky placement of game objects.
A computer opponent is instructed what to do in specific situations. This kind of instructions can make them stiff and easy to predict, thereby providing less of a challenge. Nevertheless, computer opponents can act in clever ways if much work is put into them. The downside with complex behaviour is that it requires a lot of processing power. A human opponent can act in a strategic (monotone) way and each person has different skills. However, there are more human opponents to choose from and it is easier for them to adapt their strategy in an ongoing game.

Different levels of difficulty can be provided in a game to challenge players with different skills. However, it is hard to decide what levels are appropriate since players are sensitive about their opponents (Nokia 2003a). Games that are too easy or too difficult will soon be abandoned (Nokia 2003b). Who the winner is should be unpredictable until the end and if a player put effort into the game he or she should be able to succeed.

Providing a leader board makes it possible for players to compare their results. The board can be an incentive to improve and thereby play more. However, only a few people can be on the top of a leader board, and in comparison with the best, most players look bad. Therefore, it is a good idea to compare with something else too, for example, with friends or with players of a similar skill. This gives more players a chance to excel and to have a reasonable goal to reach. Different kinds of statistics make it possible to have several leader boards. It is also possible to construct trends using existing records and time information.

2.4 Fun

Seppänen (2001) states that if a PC or console is used to play games, the game sessions tend to be long and continuous and if a mobile terminal is used, the game sessions tend to be short and frequent. Nevertheless, it can be argued that a fun game can be played for long sessions on a mobile terminal since handheld game systems have been popular for a long time; examples are Game & Watch and Game Boy by Nintendo. The length of a session is instead affected by the situation for game play, which is discussed in chapter 2.1. Mobile terminals add short and frequent sessions to the existing long and continuous ones. Another thought is whether it is fun enough to reuse old classic games. Although they are well suited for systems with limited hardware, it can be discussed whether these are the most satisfying games for a new platform (Seppänen 2001).

If the price of a game is low a player tries more game titles. However, a low price also makes it easier to discard a game that is not fun. Therefore, it is important to send the player immediately into the actions of the game (Seppänen 2001). In that way the player experiences the game from its best side and are not mislead by an introductory phase. The same idea applies when providing a demonstration version of a game.

A game becomes boring if it is played often without any new element. However, there are several solutions that stretch the life length of a game. If the
player has mastered the game it is suitable to play against other players, this challenge is further discussed in chapter 2.3. Another scenario is when the game has a static nature, for example, a small number of labyrinths. In this case, additional labyrinths or a labyrinth editor can be provided to let the players create their own labyrinths. In this way a new component is added that can stretch the life length of the game.

2.5 Current Games

A brief search on the Internet was performed to reveal what multiplayer games were available at the start of the project. The results are listed here.

The game of Chess (Network 2003) communicates with a server on the Internet. To be able to play one has to create a personal account on the server. The account can be created either using WAP on a mobile terminal or using a web browser on a PC. When an account has been created, one can connect using the game Midlet and see if there is anyone else up for a game. If there are not any tables to join one can create one and wait for an opponent. The server stores the current game data. Therefore, it is possible to play a game with seconds, hours or even days between the moves. In this case the Java files can be downloaded for free from the developers’ website (Network 2003).

The next game found was named Four-in-a-Line (Cascata 2003). It uses the short message service (SMS) to communicate between two players. The SMS is a neat way to send sparse game data. However, there is a disadvantage with using short messages for game communication, which is discussed in chapter 5.2. A one-time fee of ten dollars must be paid to download the game. Even though the targeted terminals are mobile this game is not available over the air (OTA); it has to be downloaded from the Internet to a PC and then transferred to the mobile terminal. Game features included are a computer opponent and colour display.

London based Macrospace has developed Cannons ME (Macrospace 2003), an arcade classic for 2-4 players where each player takes control of a “World War II” (Macrospace 2003) style cannon. The players eliminate each other by firing cannon balls. The idea is to specify angle and firepower of the cannon ball to hit the opponents in a windy environment. Cannons ME works with both packet- and circuit-switched connections. The targeted mobile terminals are a number of Java-enabled terminals with colour display. The game uses a server for player authentication, match making and statistics. Cannons ME is a turn-based classical style of game and is available for download at a fee of five euro.

For personal digital assistants (PDAs) running the Palm operating system, another classic game is available (Handago 2003). This game includes both colour graphics and sounds. However, the multiplayer support of this game is not impressive. It only supports an infrared connection; thereby the game is always local within meters. The game can be downloaded as a shareware version from the Internet before releasing the potential fee of 20 dollars.
3 Mobile Terminals

After motivating the use of games, the terminals available on the market was examined to find out their support for multiplayer games. This chapter provides an overview of the operating systems and programming languages available for mobile terminals and a more detailed description of the Java programming language. It also describes the implications on games from the hardware of mobile terminals, which is limited in comparison to personal computers.

3.1 Operating Systems

The support for games in mobile terminals is tied to an operating system. Two operating systems suitable for games have been identified, Symbian and Windows Mobile.

3.1.1 Symbian

In the late 1990s the operating systems of mobile terminals managed only a simple user interface and a few preinstalled applications (Wood 2002). The reason for the birth of the Symbian Operating System was that mobile terminal manufacturers (Nokia, Panasonic, Motorola, Psion, Samsung, Siemens and Sony Ericsson) saw a forthcoming need for an operating system that supports multimedia, data services, user-installed client applications and graphically rich user interfaces. The decision was to start all over, instead of patching existing systems. The reasons for co-operation between the terminal manufacturers were to create a standard operating system and to share the large investment needed (Wood 2002).

The two main user interfaces of the Symbian Operating System are: UIQ and Series 60. The UIQ uses pen-based navigation, and the Series 60 is designed for one hand use.

The common programming languages when developing games in Symbian are Java and C++. Symbian version 7.0s has support for Java Mobile Information Device Profile (MIDP) version 2.0, which includes extended support for multimedia and games, and Connected Limited Device Configuration (CLDC) version 1.0. Earlier versions support MIDP version 1.0 and CLDC version 1.0.
Chapter 3: Mobile Terminals

(Symbian 2003a). MIDP and CLDC are described in chapter 3.2.1.2 and 3.2.1.1 respectively.

3.1.2 Windows Mobile

Windows, the dominant operating system for personal computers (PCs), is available for Pocket PCs and mobile phones with Windows Mobile, which is previously known as Windows CE or Pocket PC. The source code of Windows Mobile is based on Windows CE version 4.2 (Osborne 2003).

Many of the well known Windows products available for PCs are also available in Windows Mobile. For example, it is possible to browse the Internet with Pocket Internet Explorer, send Instant Messages with MSN Messenger and play video- and music with Windows Media Player. Windows Mobile also offers access to contacts, tasks, calendar and e-mail on a PC through its synchronization with Outlook for PCs.

The native programming language in Windows Mobile is C++. Other languages, for example, Visual Basic are also supported in some terminals. Support for Java is not mentioned in the development tools-area on the Microsoft Mobile web site (Microsoft 2003a). Thus, it is not available.

3.2 Programming Languages

To be able to run a game in a mobile terminal it has to be described in a programming language. Java and C++ are the main languages available.

3.2.1 Java 2 Micro Edition

The Java 2 Platform includes three different editions; Enterprise (J2EE), Standard (J2SE) and Micro (J2ME). The Enterprise and Standard Editions offer support for web services to enable development of business applications. The Micro Edition is the platform for a wide range of limited-hardware terminals, such as mobile phones and PDAs. The J2ME architecture defines configurations, profiles and optional packages. The configurations are composed of a virtual machine and a minimal set of class libraries. These configurations must be combined with profiles in order to provide a complete runtime environment (Sun 2002).

3.2.1.1 Configurations

A terminal that has the ability to connect to the outside world is called a connected device, it does not need to be wireless. The J2ME consists of two
configurations for connected devices, the Connected Limited Device Configuration and the Connected Device Configuration.

The Connected Limited Device Configuration (CLDC) is the result of a collaboration of companies representing different industries. The specification of CLDC consists of three parts; a virtual machine, a small subset of the classes in Java 2 Standard Edition (J2SE) and some new application programming interfaces (APIs) for in- and output (Giguère, E., 2003).

The virtual machine (VM) used for mobile terminals is named Kilobyte Virtual Machine (KVM). The name comes from the fact that the KVM runs on terminals equipped with only tens of kilobytes of available memory, the lower boundary is approximately 60kB (Muchow 2002). A limitation of the KVM is that floating-point types are not supported. The reason for this lack of support is that almost no CLDC-terminal has hardware support for floating-types and the cost of supporting it in software was considered too high. Another limitation concerns error handling. Recovery from error conditions in embedded systems is usually terminal specific and handling all of the error classes in J2SE would impose a major overhead on the implementation, given the strict memory constraints in CLDC-terminals (Sun 2000a).

The subset of classes from J2SE only includes selected classes from the three packages `java.lang`, `java.io` and `java.util`. However, an additional package is introduced: `javax.microedition.io` (Taivalsaari 2003). This package also includes new APIs for in- and output called the generic connection framework (GCF). This framework provides abstraction of seven different types of connections (Giguère 2001). Thus, in CLDC the specification is done, but not the implementation. The classes that provide the implementations are on the profile level, for example, in the Mobile Information Device Profile (MIDP).

The other configuration, Connected Device Configuration (CDC), is intended for wireless terminals with at least a few megabytes of available memory. The VM is a full-featured runtime environment without restrictions. This configuration will neither be examined further nor used since terminals with the CDC is part of delimitation 2 in chapter 1.4.

### 3.2.1.2 Profiles

Profiles are basically a set of high level APIs. The most common profile, used together with CLDC, is the Mobile Information Device Profile (MIDP). It is possible to use more than one profile at a time if both the VM and the terminal support them.

MIDP defines the Java application environment for mobile information devices. An application running in the MIDP environment is called a Midlet. There are currently two versions of the MIDP specification, version 1.0 and version 2.0.
MIDP 1.0 was the first profile for J2ME; it was released in the year 2000. All MIDP implementations must provide support for accessing HTTP version 1.1 servers and services (Sun 2000b). The ability to download an application to a mobile terminal over-the-air (OTA) is only a recommended practice, found in a separate document (Pratt 2003).

The hardware requirements of MIDP 1.0 include a display size of no less than 96x54 pixels. All colour depths are acceptable.

The network security functions in MIDP version 1.0 are limited and forces developers to create their own. Java applications run in a so-called sandbox, a protected environment. For example, it is not possible to access the file system of a terminal from within an application.

MIDP 2.0 was released in November 2002; it includes updates and new capabilities. MIDP 2.0 is backwards compatible with MIDP 1.0 (Pierce 2002). Midlets can now use the HTTPS protocol, which enables encryption and endpoint authentication besides the functionality of the HTTP-protocol (Knudsen 2002b). These functions are necessary for secure handling of sensitive information like credit card numbers and passwords (Appnel 2002).

Another new feature is the push registry that allows a Midlet to respond to incoming network connections. Through a class named PushRegistry a Midlet can register an inbound listener. This listener is active even when the Midlet is not running. When the listener receives an incoming request, the Midlet is launched and the request is passed to it.

Audio support is also added, which enables a Midlet to play audio without the need of an optional package. The API supports tone generation, tone sequences and, if the terminal supports sample audio, WAV-files (Appnel 2002).

The new Game API adds features for games animation and key presses. Sprites can be used to keep track of animations and collisions between sprites can be detected. Graphic related additions include layers, an off screen buffer and direct access to image data. Several key presses can be detected simultaneously (Nokia 2003h), the previous functionality is described in chapter 3.2.1.4.

The MIDP 2.0 specification also includes over-the-air (OTA), the former recommended practice for Midlet download. Thus, all terminals with MIDP 2.0 support can download and install games and other applications OTA.

A new security model introduces the concept of trusted and un-trusted security domains (Pratt 2003). If the Midlet belongs to an un-trusted domain it will have the same restricted access as with MIDP 1.0. The Midlet is un-trusted as long as it has not been signed by a trusted source. When signed and thereby trusted, it has access to more sensitive and restricted functionality in the mobile terminal.
3.2.1.3 Optional Packages

An optional package is like a profile, a set of APIs. The difference is that it does not contain a complete application environment. An optional package is always used in conjunction with a profile and a configuration (Giguère 2002).

A terminal manufacturer can provide optional packages to allow application developers to use functionality in a terminal that are not included in the MIDP specification. For example, some terminals have access to the list of missed calls, SMS functionality and the camera. If an application developer uses an optional package, the application source code has to be modified to make it work on a Java-enabled terminal that lacks the optional package (Nokia 2003b).

3.2.1.4 Application Programming Interface

An Application Programming Interface (API) is used by application programmers when implementing applications. It defines which ready made classes are available to the programmer. Some important parts of the J2ME API are described here.

A mobile terminal requires a display to present all the action and events in a game. Accordingly, every game Midlet has a reference to a Display object. This reference can be used to retrieve information about the real display, for example, the number of colours and the display size. In addition, the Display object controls what is visible on the display - and when.

There are a number of graphical components in MIDP that are ready to use. Four of these components are used in conjunction with the Screen class. Screen itself is not displayable; it is a parent class for components with different functionality that are displayed on the display. The four visible components are TextBox, List, Alert and Form. The first three have exclusive right to the display.

The TextBox class can, for example, be used for typing in a name when a high-score has been achieved. It can hide text behind stars (*). Thus, passwords can be protected when they are typed in.

With the List class, multiple choice-components are created; they can be presented in one of three different formats: multiple, exclusive and implicit. Using the format multiple (check boxes) it is possible to select several options. This format is a good choice for the options menu of a game. The exclusive format (radio buttons) can be used when selecting degree of difficulty. Finally, using the implicit format generates an event when an option is selected. The event can for example, be used to jump between menus.

The Alert class is used to notify the user of an event. The event can be anything from an informational text to an error message.

The last visible component of Screen is Form. The Form class is a little different from the other three since when using a Form, it is possible to show multiple components on the display at once. If the components in a Form do not
fit within the display, scrolling up and down is possible. There are six different components that can be added to a Form; ChoiceGroup, ImageItem, TextField, DateField, Gauge and StringItem. The ChoiceGroup can be used for selecting an opponent on a game server, the ImageItem to view a picture of him or her, and the TextField for in-game communication – for example, a chat.

For some applications the available components under Screen is not enough; customized components need to be created. The customized components are made using the classes Canvas and Graphics.

Game developers will make extensive use of the Canvas class. The Canvas itself is just like a white paper, which can be painted using different shapes and colours. Everything drawn on the Canvas will be visible on the display. To draw something on the Canvas, one has to use a Graphics object. Over 30 methods are included to draw shapes and text, set fonts and colours, and much more. However, the font support in J2ME is poor compared to J2SE. MIDP 1.0 does not support the use of layers. Thus, methods with fill operations are used to colour the background.

MIDP 1.0 does support sounds. Using the AlertType class, five predefined sounds are available, they are tied to various conditions; alarm, confirmation, error, info and warning. Several mobile terminal vendors have API extensions to their terminals, which makes it possible to for example, create customized sounds with a certain frequency and duration. These extensions enhance the ability to develop feature rich games with suitable sounds.

MIDP 1.0 does only support event handling for one key at a time. For example, in a football game this limitation makes it impossible to run and shoot simultaneously. However, this problem is resolved in MIDP 2.0, which is described further in chapter 3.2.1.2.

There are two sets of constants to use when referring to a pressed key, Key Codes and Game Actions. Key Codes are mapped directly to the keys on a mobile terminal. Some of these Key Codes are guaranteed to be available on any MIDP-terminal that uses the International Communication Union (ITU-T) keypad standard. The standard includes the numeric keys 0 to 9, star (*) and pound (#). To write portable games it is recommended to use those keys. Examples of keys that cannot be guaranteed to work are the directional keys, which are located just below the display.

The other set of constants includes the names UP, DOWN, LEFT, RIGHT and FIRE. These words are referred to as Game Actions and facilitate event handling for game related events (Muchow 2002). Since the layout of keypads differ, a user interface that is suitable for one keypad layout may not be suitable for another layout. The idea behind Game Actions is to let the terminal manufacturers map common game actions to appropriate buttons on each terminal model (Nokia 2003a).

A Game Action can be assigned one or more Key Codes. For example, if a terminal has directional keys below the display, the one pointing upward and the numeric key 2 can be assigned to the Game Action UP. Pressing any of those keys
will invoke the method for the up-action. A nice feature in a game is to let the player select his or her key mappings. This principle is used in many of the popular games for personal computers today.

The methods for event handling are keyPressed and keyReleased. One only needs to examine the current Key Code to determine which Game Action was generated.

As mentioned in chapter 3.2.1.1, there is a framework called Generic Connection Framework (GCF). This framework includes one class and seven connection interfaces. The purpose with GCF was to define abstractions for networking and file I/O as general as possible. The manufacturers select which interfaces they want to implement, depending on the terminal’s capabilities, and thereby which connection-types that should be supported. A basic implementation of the MIDP only supports HTTP since a MIDP-enabled terminal must be able to communicate with a web server. Only terminal-initiated communication is possible to a server. Thus, the Midlet constantly has to poll the server for new game status. Unfortunately, polling is not an efficient method. Furthermore, HTTP carries a great deal of overhead to each request.

The class that supplies the HTTP connectivity is called HttpConnection. This class is available on every MIDP-terminal. An InputStream object is used for reading the contents from a web server and an OutputStream object is used to send a request to a web server.

3.2.2 C++

Games written in the C++ programming language does in many ways have the same possibilities as games written in Java. However, C++ has better performance in memory use and execution speed (Symbian 2003b) since C++ games have been compiled to machine code before execution. Java applications on the other hand execute in a virtual machine that creates machine code at run time, which is slower in theory.

C++-games can access resources external to the game in the directories of the file system and also control hardware, for example, the camera. However, some functionality of Java is omitted in C++, for example, the exception handling (try, catch and throw); instead the Symbian OS provides its own exception handling mechanisms (Symbian 2004a).

Synergenix (Synergenix 2003d) have developed a software-based game engine for mobile terminals, called Mophun. It is targeted solely on games and includes for example, rich graphics support. Registered developers have access to an Application Programming Interface and a Software Development Kit. The supported programming languages are C, C++ and Assembler (Synergenix 2003f). To run Mophun enabled applications a special virtual machine has to be installed in the terminals.
Synergenix want to control which games are developed. Therefore, before testing a game on real hardware it has to be certified by Synergenix. Although this procedure ensures the quality of the games, it also gives Synergenix a control of every game and its contents. In addition, an agreement has to be signed with Synergenix (Synergenix 2003b) to guaranty revenue for the software developer and a commission fee for Synergenix. For these revenue streams to be generated the games have to be charged at download time. To prevent illegal copying, each game copy is locked to a single terminal.

3.3 Hardware Capabilities

To know what game types are appropriate for mobile terminals, it is necessary to consider the targeted hardware. Games that work on personal computers are not always easy to translate to mobile terminals, which have different hardware capabilities.

An obvious limitation of mobile terminals is their small displays. This limitation makes it hard to present lots of information at the same time on the display. However, a large game world does not fit into any display. A solution to this problem is to scroll the game world and show only a small part at a time (Nokia 2003f). The portrait format of most mobile terminal displays makes side-scrolling games a bit tricky. The character comes closer to the edge of the display and therefore the game has to go slower to allow the player to react to new objects in its path. Otherwise, the player has to react faster than if a display with a landscape format was used (Nokia 2003a).

It is important for games to utilize the whole display of a mobile terminal since the displays are small to begin with. The principle of using the whole display versus using a part of the display is illustrated in Figure 17.

To suit different display sizes the graphics can be stretched dynamically. However, the best looking result is to create new graphic resources for each display size (Nokia 2003a). This procedure is of course more time consuming. To make good proportions, a main character in a game should use between 10 and 15 percent of the display size in each direction (Nokia 2003a). Even though the resolution will improve in future terminals, it is likely that the display dimensions will stay small to benefit mobility.

As described in chapter 3.2.1.2, the sound capabilities varies between different terminal models and if a terminal can produce sounds it is often limited to simple tone sequences. Since mobile games can be played anywhere, it is important to remember that in-game sounds can disturb other people. Thus, it is necessary that all sounds can be turned off (Nokia 2003a). On the other hand, if the environment is noisy, subtle sounds are not heard (Nokia 2003b). As a result, games for mobile terminals should not depend on sounds but use them as enhancements.

Video calls, a voice call that simultaneously use a camera to display the person at the other end, drains batteries at a high rate (Danielson 2003). To
compare, games also use graphics and sounds. Another factor that drains the battery is extensive use of the processor, which is common in games (Nokia 2003b). Nevertheless, without further studies it is difficult to exclude certain game types because of the battery capacity, since all games are playable for some amount of time, although short game sessions are preferable. If the capacity of batteries improves to support video calls, games will also benefit from it.

The memory used for storage in terminals has to be large enough to accommodate the source file of a new game. Note that the storage area for games is also used for other storage like images and ring tones. Therefore, some terminals use memory cards to provide extra storage. Another concern for some terminals is the heap size. Objects created at run time need memory, often in excess of the game file size (Nokia 2003a).

Text input is slow and difficult on mobile terminals since the keypad is constructed for typing numbers (Nokia 2003b); a typical keypad is displayed in Figure 3. Several characters are often located at each button, which results in frequent typing errors. In a game with text commands people do not like when a word is discarded because of a typing error (Seppänen 2001). Therefore, it is logical to avoid typing as much as possible. Instead, a list of predefined messages should be used when the possible answers are known beforehand.

In games for personal computers, the use of a mouse as pointing device is common. However, pointing devices are rare for mobile terminals. An exception is terminals with the UIQ-interface (pen-based) available for the Symbian operating system, for example, Sony Ericsson P800. PDAs also use pointing devices. Without a pointing device it is difficult to select objects in a playfield and to change view quickly. A pointer can still be used if it is moved with the keys or with a joystick as in the Nokia 7650 and Sony Ericsson T610 terminals, but then it is difficult to move the pointer with both speed and accuracy. A better approach can be to lock the pointer to specific objects, like in sequential menus. This principle is well proven in console games since those systems normally do not have a pointing device.

![Figure 3: A typical keypad on a mobile phone.](image-url)
Current Nokia Series 60 terminals have a central processing unit (CPU) with a clock frequency of 104 megahertz (Nokia 2003a). This number is small compared to the CPUs in personal computers. Most of the processing power for games is used for graphics, to calculate how to update the display. However, since the game files are small, the amount of graphics included in a game is constrained. In addition, a small display contains few pixels, which require less processing power than a large display with many pixels (Nokia 2003a). Therefore, the processing power found today should not be a constraint for current game types (Nokia 2003a). The future use of 3D-graphics will require more processing power.

The position of a mobile terminal can be decided if it supports the Global Positioning System (GPS) or if the mobile network has built in positioning services. This feature can be used in games; an example is the Botfighters positioning game (TeliaSonera 2003a). The game idea is that the competitors can shoot each other, using their mobile terminals as a weapon, when they come close enough in the real world.
4 Network Characteristics

This chapter describes mobile networks; GSM, GPRS and UMTS are covered. The internal structure of the networks and the important characteristics are investigated.

4.1 Connection Modes

Two connection modes are available for data networks, circuit- and packet switched data. They are described and their differences are highlighted.

4.1.1 Circuit Switched Data

Circuit switching, designed in 1878 (Copeland 2000), is the technique that dominates in the public switched telephone network (PSTN) today. The basic principle is to create a connection by reserving network resources between two terminals. These reservations prevent others from using the link – even if it is inactive.

When using circuit switched data, the information arrives at the destination in the same order as it was sent. Disordered arrival of information is bad when dealing with voice-data since it introduces a delay (Farley 2004).

Payment is often based on connection time, which implies that persistent connections are expensive if the time between transmissions is long. If intermittent connections are used instead, setup delays are introduced since the network resources have to be reallocated for each transmission. Furthermore, if only short bursts of data are sent, the setup time represents a large part of the total time connected (IEC 2000).

4.1.2 Packet Switched Data

An example of a packet switched data network is the Internet. The principle is: large chunks of data to be sent are divided into standardized packets with header fields and payload. Examples of header fields are: source and destination addresses, payload size, sequence number and error-checking information.
In packet switched data, the packets can use different routes to reach the destination. Thus, the data can arrive disordered. The receiver restores the original data by looking at the sequence number in the packet headers.

A disadvantage of packet switched data is that packets can arrive to a network node in a higher rate than the packets are routed, and then the packets will be queued. If the size of the queue is exceeded, packets will be lost (IEC 2000). When demand exceeds supply on a network node, congestion is introduced.

There can be packets from different sources on the same link, which allows users to share the available bandwidth since network resources are reserved only when needed for transmission. Thus, it is possible to maintain an idle connection without blocking any resources. The most common methods of payment are by the amount of transferred data and flat rate – a monthly charge independent of the amount of transmitted data.

4.2 GSM

The global system for mobile communication (GSM) is a circuit switched data network that uses Time Division Multiple Access (TDMA). When a user connects to the GSM-network, a certain time slot (described in section 4.3.1) is dedicated for the entire session. The transmission rate is limited to 9.6 or 14.4 kilobits per second and time slot, depending on the support in the terminal and the operator (Liljestam 2004). If the terminal supports it, one can choose to use high speed CSD (HSCSD). With HSCSD, several time slots can be allocated for a single terminal.

A GSM-network consists of several nodes and can be divided in two parts (see Figure 4); an access-network and a core-network. The access-network, also known as a base station subsystem (BSS), consists of base transceiver stations (BTSs) and base station controllers (BSCs). The BTSs handles the radio signals to and from the mobile stations (MSs). The BSCs handles radio resources and control handovers. A BSC can manage several BTSs.

The core-network consists of mobile switching centres (MSCs), gateway MSCs (GMSCs) and a number of databases. The MSCs controls several BSCs and is responsible for the routing of the calls, the tracking of the mobile users and security functions. Connections to and from fixed networks, like the public switched telephone network (PSTN), are handled by the GMSCs. The databases are connected to the MSCs and holds information about, for example, the position of a user, bills and rights.
4.3 GPRS

General packet radio service (GPRS) is a packet switched data network. In GPRS, a time slot is reserved only when data is transmitted. After transmission, reserved time slots are immediately freed to serve other users. When no transmission takes place, users stay online but network resources are not blocked. The software in the GPRS nodes controls the maximum number of online users.

GPRS was designed to run on top of the existing GSM-network; software has to be upgraded and hardware has to be added. The upgrades concern the software in the MSCs and the BSSs. The hardware consists of GPRS support nodes (GSNs) and a packet control unit (PCU).

Two new types of GSNs are introduced forming the IP-based GPRS backbone as displayed in Figure 4. The nodes are called Serving GSN (SGSN) and Gateway GSN (GGSN). The SGSN is on the same hierarchical level as the MSC and has corresponding tasks. The MS makes a GPRS attach when it is turned on or told so to establish a logical link to the SGSN. An attachment has to be done in order to use the GPRS-services. The GGSN is responsible for the connection with other packet-based networks, such as the Internet, and stores information about which SGSN the users are connected to. In this way, the GGSN knows to what SGSN to route incoming packets from other networks. The PCU is responsible for the GPRS packet data radio resource management in BSS and enables GPRS users to share a time slot (Krantz 2004).

![Figure 4: Distribution of the components in a GSM- and GPRS-network.](image-url)
GPRS uses the same access-network (BSSs) as GSM does. The difference lies within the routing of data from the BSC. Instead of sending data to a MSC, the BSCs send data to a SGSN. Otherwise, the GPRS-network is only communicating with the GSM-network via SS7-signalling (Mayer 2004). Protocols (2004) provide more information about SS7-signalling.

To sum up, the characteristics of GPRS are:

- possibility of high bandwidth
- effective utilization of the network- and radio resources
- often volume based charging
- seamless integration with the Internet
- flexible co-existence with the GSM-network and voice traffic
- possibility to stay online at no extra charge
- often lower priority than voice- and GSM-traffic

4.3.1 Time Slots

The available frequency spectra for GSM and GPRS are divided into channels by using a technique called Frequency Division Multiple Access (FDMA). The channels in turn are divided into time slots by using a technique called Time Division Multiple Access (TDMA). The time slots hold the transmitted data. In GSM, a time slot is permanently allocated for a particular MS during the entire call as displayed in Figure 5 and Figure 6. The allocation in GPRS allows several users (up to eight) to share the same time slots, which is illustrated in Figure 7. Depending on the load of traffic on the BTS, a single MS can also be assigned several time slots (multi-slot operation).

![Figure 5: A user connected with CSD reserves one time slot during the entire call.](image)

The allocation of time slots is dynamically made in the BTS. Uplink and downlink time slots are allocated separately, which can result in asymmetric data traffic. In theory, an assignment of eight time slots for a MS is possible if the BTS has no other traffic. In practice, the number of time slots assigned is controlled not only by the BTS, but also by a limitation of the MS – the multi-slot class, which is described in section 4.3.2.

![Figure 6: A user connected with HSCSD can reserve more than one time slot.](image)
The operators have the possibility to set different priority on voice and data traffic. For example, if a voice call has higher priority than a data transfer, a time slot used by a GPRS-user will be handed over if a GSM-user needs a slot.

The availability of time slots also depends on the number of BTSs in the area. An 1800/1900 megahertz network requires more BTSs than a 900 megahertz network does because of the fact that higher frequencies result in shorter range. More BTSs means more time slots and less people on the same BTS (Hess 2002).

4.3.2 Multi-slot Classes

There are a total of 29 different multi-slot classes defined in the GPRS-standard. The first 12 of these classes are the interesting ones since they are frequently used in today’s terminals. The class definitions specify three important factors concerning bandwidth, namely:

- the number of concurrent time slots for uploading data
- the number of concurrent time slots for downloading data
- the total number of concurrent time slots

For example, a terminal with the multi-slot class 10 can handle four time slots for downloading, two time slots for uploading and a total of five concurrent time slots as shown in Table 1. Thus, using four receive- and two transmit time slots concurrently are not possible. The terminal will then dynamically choose the combination of time slots to use, 3+2 or 4+1, depending on whether data is requested or sent (Hess 2002).

The multi-slot class is important to think about if buying a terminal for a specific purpose. For example, if much data is to be sent, a terminal that supports two uplink time slots double the uplink bandwidth.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 7: Two GPRS-users share the same time slots.

Table 1: The first 12 multi-slot classes and their time slots.
4.3.3 Channel Coding

Channel coding introduces redundancy into the data packets for detecting and correcting transmission errors. GPRS offers four different types of coding schemes (CSs), listed in Table 2. The packets sent include payload and coding. A robust coding, for example, CS-1 in Table 2, uses more coding bits, which results in less payload bits per packet and thereby a reduced bandwidth. If the operator supports more than one CS, the choice of CS depends on the quality of the radio link between the MS and the BTS. Under noisy conditions, the network can use CS-1 to ensure higher reliability.

Table 2: Data rates for Coding Scheme 1 to 4.

<table>
<thead>
<tr>
<th>Coding Scheme</th>
<th>1 Time Slot [kbps]</th>
<th>4 Time Slots [kbps]</th>
<th>8 Time Slots [kbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS-1</td>
<td>9.05</td>
<td>36.2</td>
<td>72.4</td>
</tr>
<tr>
<td>CS-2</td>
<td>13.4</td>
<td>53.6</td>
<td>107.2</td>
</tr>
<tr>
<td>CS-3</td>
<td>15.6</td>
<td>62.4</td>
<td>124.8</td>
</tr>
<tr>
<td>CS-4</td>
<td>21.4</td>
<td>85.6</td>
<td>171.2</td>
</tr>
</tbody>
</table>

4.3.4 Bandwidth

The achieved bandwidth in GPRS depends on several factors. Three of the most important factors have been described earlier and they are:

- **Coding Scheme** - If the operator supports different coding schemes, the radio conditions will decide what CS to use.
- **Time Slots** - The number of time slots assigned depends on the load of traffic and multi-slot priority on the BTS.
- **Multi-slot Class** - The multi-slot class on the terminal will decide how many concurrent time slots that can be used.

Another factor that affects the bandwidth is the amount of packet loss and delays.
4.3.5 Latency

There are many reasons why a delay can be introduced in a network. As a user, it is hard to affect these, since they are introduced within the networks. The best thing one can do is to get a clear view to the BTS.

A good measurement on the quality of a connection is the round-trip time (RTT). This measurement reveals the time from a transmission of a request until the answer has been received. Factors that affect the RTT are:

- **Packet Loss** - If a packet is not received in a reasonable amount of time it can be considered lost. However, control mechanisms can be used to guaranty the reception of a packet. One solution is to request that the lost packet is resent. It must then traverse the network again, which adds a delay.

- **Resource Allocation** - The allocation of a time slot takes about 800 milliseconds. After 200 milliseconds of network inactivity, the time slot is released (Lundqvist 2004). If a small amount of data is transmitted, the time slot allocation takes a large portion of the total transmission time.

- **Jitter** - Jitter is the unwanted variation of time delays that can be introduced when using packet switched networks to transport data (Mossé and 1996). Jitter is more or less noticeable depending on the type of information received. For example, jitter is less noticeable when receiving e-mail than streaming media, since jitter results in a jerky playback of the stream. Jitter compensation buffers can be used to overcome this undesirable effect (Bashandy 1999). However, buffering increases the time until the streaming media can be presented.

- **Handovers** - When the terminal switches BTS, for example, when another BTS is closer to the terminal, it is called a handover. The conditions at the new BTS can be worse than at the previous, for example, it can be more occupied. Even the handover process itself can cause delays and data losses (Yu 2003).

- **Link Blackout** - A radio coverage disruption can occur when entering a tunnel or moving far away from a BTS (Yu 2003). When the radio coverage ceases to exist during a long period of time, it is called a link blackout.

- **Delivery Delay** - In order to know which SGSN the GGSN shall route incoming packets to, the MS periodically has to update its location. If it does not update its location, the MS has to be paged (searched for), which results in a delivery delay (Dhananjay 2004).
4.3.6 Terminal Classes

There are three types of classes defined for terminals; A, B and C. With a Class A terminal, GPRS-services can be used at the same time as GSM-services. A Class B terminal can be attached to both GPRS- and GSM-networks, but only one of them can be used at a time. A Class C terminal cannot be attached to both networks concurrently; it has to switch between them in order to use it. Exceptions are SMS messages, which can be sent and received at any time (Dhananjay 2004).

4.3.7 Service Access

In order to access GPRS-services, the following criteria have to be fulfilled:

- a GPRS-enabled terminal
- a telephone network that supports GPRS
- a subscription for GPRS-services

4.4 UMTS

The first generation (1G) of mobile networks is the analogue Nordic Mobile Telephony (NMT). The digital GSM and GPRS represent the second (2G) generation and the new Universal Mobile Telecommunications System (UMTS) is the third generation (3G). UMTS represents an evolution in terms of services and bandwidth. UMTS is a multimedia network that enables video-calls and music streaming with transmission rates initially up to 384 kilobits per second and in later releases up to 14 megabits per second (Ericsson 2004).

Like GPRS, UMTS is a packet switched data network. However, the TDMA technique with time slots which is used in GSM and GPRS is no longer used. Instead, UMTS uses Wideband Code Division Multiple Access (WCDMA) as radio access technology, which means that all users transmit at the same time but each session is identified with a unique code. The new radio access technology means that a new radio access network (RAN) has to be built. UMTS uses a higher operating frequency than GSM, around two gigahertz, which means that the range of the radio waves is shorter. As a result, more base stations are needed to cover the same area (TeliaSonera 2003b). Figure 8 shows the new nodes in UMTS RAN.

The basic core network architecture for UMTS is based on GSM network with GPRS. The UMTS RAN displayed in Figure 8, and the GSM/GPRS BSS are both connected to the GSM/GPRS core network. The GSM BSC corresponds to the UMTS Radio Network Controller (RNC), which for example, handles channel allocation, handovers and controls the radio resources. The Radio Base Station
(RBS) handles for example, air interface reception and transmission, like the BTS in GSM.

Figure 8: The new radio access network of UMTS is connected to GSM and GPRS core networks.
5 Multiplayer Business

This chapter covers the present state of the multiplayer games business. It describes the availability of components needed for multiplayer games, for example, the terminals. The business parties are also described along with the factors that generate revenue.

5.1 Present State

Not many multiplayer games were found in the Internet search in chapter 2.5. The reasons why those games are rare can either be technical or economical. The mobile network characteristics are examined in chapter 4.

According to analysts, 80-90 percent of the mobile game revenues in the world are collected in Japan and South Korea (Nokia 2003e). However, Nokia believes that the market for mobile games will grow outside Asia with the introduction of their Series 40- and Series 60-terminals, estimated to more than 50 million terminals at the end of 2003. Furthermore, Nokia believes that investments in mobile games are not likely to generate revenues in the short term, but it can be an investment in future growth (Nokia 2003e).

![Figure 9: Current distribution of game fees in Europe.](image)

In Europe, application developers are given only a one-time income, when game copies are sold (Synergenix 2003c), which is illustrated in Figure 9, and that income is used to cover development costs. Since the game developers are not
given a share of the network traffic fees (Nokia 2003f) there is no incentive for them to provide after-sale support or to host game servers. As a consequence, mostly single player games are developed (Nokia 2003f).

5.2 Necessary Infrastructure

If TeliaSonera should enable multiplayer games three components are needed: a distribution system for the game clients, a game server and a billing system.

The old way to supply games was to embed them before shipping a terminal. This approach limits the target group to those with a specific terminal model, which prohibits a multiplayer game session with a friend that use another terminal model. The new way to supply games is to let a user download games over-the-air (OTA) to a terminal. This delivery method makes a game truly mobile since it can be found when browsing a WAP-site, downloaded OTA to the terminal and game play is online through a mobile network. The download and game server can be combined and also handle auxiliary functions, for example, statistics. However, for large scale deployment it is better to separate the servers to avoid that download interfere with game play.

Not all communication principles are equally suitable for games. The Short Message Service (SMS) is currently priced for exchanging just a few messages, thus a single game can get expensive if many messages are exchanged. Infrared- and Bluetooth-links only offer local gaming and do not utilise any mobile network, therefore they will not be considered further. However, Bluetooth-links can have shorter latency than mobile networks. Today a game server is needed to enable online play since Java-applications in mobile terminals cannot connect directly to other mobile terminals. The game server can either administrate in-game information or manage a game state, which is discussed in chapter 6.7.9.

A billing system is needed to charge for the game service. A new service will not be given a high priority if it cannot be charged for (Ericsson 2003). Different ways to charge for a service is discussed in chapter 5.6.

5.3 Terminals

To run a game in a mobile terminal, the terminal must be able to execute the code describing the game. Furthermore, the appropriate terminals must be available on the market.

5.3.1 Requirements for Java Games

All games that have to be installed in a terminal require the terminal to support applications written in the specific programming language. Otherwise, the game
cannot be run. This information is available in the terminal specifications. For Java games, if optional packages are used several versions of the game has to be provided since the support for optional packages varies between terminals. Consequently, it is important to consider the use of optional packages before a game is implemented. Some of the Java-games provided by TeliaSonera (TeliaSonera 2004b) are restricted to only a few terminals, for example, Siberian Strike (Gameloft 2003).

The type of communication used is important for multiplayer games. For example, a SMS-based game, without installation in the terminal, can be played with every terminal that supports the Short Message Service (SMS). As noted in chapter 3.2.1.2, all Java-based games support the HTTP-protocol which can be used to transfer in-game data over a suitable connection, for example, GPRS.

5.3.2 Availability

Statistics for quarter two 2003 (Smith 2003) shows the market share of the largest handset manufacturers. The statistics is based on the number of shipments during that period. The world leader is Nokia with a share of 34.6 %. The rest of the top five is Motorola 13.4%, Samsung 10.1 %, Siemens 6.8 % and Sony Ericsson 5.7 %. Figure 10 illustrates these proportions.

In September 2003, the market leader Nokia has 30 terminals with Java support (MIDP 1.0) (Nokia 2003g). Motorola introduce a few terminals; all support Java except one that uses the Windows Mobile operating system. Samsung, Siemens and Sony Ericsson also have several models that support Java (Sun 2004). Eight mobile terminals use the Symbian OS (Symbian 2003c). They currently come from four different manufacturers, among those are Nokia and Sony Ericsson, but releases from Samsung and Siemens will come in the near future. In September 2003, only Sony Ericsson and Nokia terminals have support for the Mophun game engine (Synergenix 2003e). In March 2004, Sony Ericsson declared that they will abandon Mophun and concentrate on games written in Java (Ryberg 2004).
As of February 12 2004 four terminals supporting Java MIDP 2.0 is available on the market (Motorola A760, Nokia 6600, Sony Ericsson P900 and P908) (Sun 2004). Naturally, it will take some time until these terminals are widely spread, until then it is probably better to stay within the MIDP 1.0 specification.

Motorola have recently launched their first Symbian-based terminal, but they have also decided to sell-off their part from the company that licences the Symbian OS. It is unknown if they do this for strategic or economic reasons (Berniker 2003). However, Motorola have teamed up with Microsoft to deliver a series of terminals equipped with the Windows Mobile operating system (Microsoft 2003b). Nevertheless, they can still launch Symbian-based terminals but their interest may have weakened.

So far only a few Windows-based mobile terminals are available (Dustin 2003). However, with many popular software titles in-house and a recent signing with Motorola, the potential is huge.

5.3.3 Easiness of Game Installation

An interesting statement is “[t]he customer is solely a terminal user: without much computer literacy to install software using PC [and] no patience to keep waiting during software download” (Tsang 2002). Installing new software from a PC removes the flexibility of a mobile terminal since it requires access to a PC with an Internet connection and means to transfer files from the PC to the mobile terminal. Consequently, over-the-air (OTA) provisioning should be used for commercial deployment of Midlets (Nokia 2002b). Note that the MIDP 1.0 specification does not require OTA to be supported by a terminal.

Typing a URL with a numerical keyboard is hard and time consuming as described in chapter 3.3, therefore it is important to publish games using hyper
links, for example, on WAP-sites, to avoid the need for the user to type a URL to find the game files.

To get a hands-on experience of OTA-provisioning, a game has been downloaded a few times from the game server with GPRS. The duration of the download and automatic installation of The Right Way was measured to 45 seconds (file size 60 kilobytes); the installation part completed in two seconds. The measured 45 seconds felt like a long time, thus it is valuable to minimize the file size for applications delivered OTA. Larger files can take several minutes, which may be too time-consuming to download OTA.

5.4 Customer Interest

The interest in mobile games is related to a user’s historic interest in games and their current needs. As mentioned in chapter 2.1, it is suitable to play a game when waiting on transportation. This need can be made visible to a user with help from a marketing campaign or simply by viewing others having fun playing.

A strategy to change customer behaviour is to provide a service for free during a limited period of time. An example of this strategy was the TeliaSonera MMS-campaign in December 2003 where the new Multimedia Messaging Service (MMS) could be used for free. For a multiplayer game this could mean free network play or free game download. Another similar strategy that is suitable for expensive products is to provide a demonstration version. In that way, a user is able to try and evaluate a new product or service before making the investment. If a multiplayer game generate most of its revenue by continuous game play, the one time fee that is charged for single-player games can be reduced or removed and thereby encourage users to try new games.

If a multiplayer game is included in a service collection like TeliaSonera Go (TeliaSonera 2004a), it can be used as an incentive for customers to choose a specific network operator (Synergenix 2003a). Other positive effects from multiplayer games can be increased use of data services and customer loyalty (International 2003).

5.5 Business Parties

If a network operator does not have the ability to develop games they need a partnership with a content provider (Ericsson 2003). A content provider wants to be paid even if no revenue is collected from the customer (Ericsson 2003), to cover the development costs. Until revenue is generated it is costly for a network operator to provide many games.

For multiplayer games an additional source of revenue comes from continuous game play. In Asia network operators share the traffic fees generated from
multiplayer games with content providers (Nokia 2003e); this principle is illustrated in Figure 11.

Either a content provider or a network operator can host the necessary game servers. However, a network operator may want to reserve game play for its customers, as discussed in chapter 5.4. Some content providers do not have means to distribute games or to charge for them; in this case the network operator can provide these services and get additional revenue (Ericsson 2003), this principle is illustrated in Figure 12.

---

**Figure 11:** The traffic fee from a multiplayer game can be shared between a network operator and a game provider.

**Figure 12:** Providing distribution services can increase the revenue for a network operator.
5.6 Charging Methods

When pricing a new service it is important to match the user perceived value (Ericsson 2003). If the monthly bill is larger than expected due to excessive game play, a customer is lost (Nokia 2003f). There are mainly three charging methods available, content-, service- and bearer charging (Ericsson 2003).

TeliaSonera use volume-based prices when charging for GPRS-subscriptions, for example, a price per transferred kilobyte. This method is called bearer charging and the income for the network operator comes only from providing network access. However, for a multiplayer game the network is utilized both when downloading the game files and during game play. When using this charging method, it can be hard for a customer to understand what he or she is charged for and how high the total cost will be (Ericsson 2003).

Service charging often means a unit price and is used for person-to-person messaging and positioning services (Ericsson 2003). To make the pricing easy to understand the same price is used regardless of the amount of data that is actually sent, thus the bearer charging is shut off. For example, this principle is used for the Multimedia Messaging Service (MMS).

According to Ericsson (2003), content charging should be the preferred charging method for games and other new services. The users are charged for the experience or value instead of the technology, for example, a one-time fee when a game is downloaded or a monthly subscription (Ericsson 2003). In that way the users understand the price they are paying and it corresponds to the value they receive. According to the International Game Developers Association (International 2003) it is becoming more common with subscriptions and pay-for-play solutions. Subscription based pricing is suitable when the users cannot control the amount of information sent, for example, when receiving a notification for each goal in a football game (Ericsson 2003).

5.7 Revenue from Games

As noted in chapter 5.1, single-player games generate a one-time income when they are sold (Synergenix 2003a). The factors that influence the revenue from single-player games are the unit price, the target group and partnerships with content providers (Nokia 2003e). For multiplayer games an additional source of revenue comes from continuous game play, this is discussed further in chapter 5.5.

The part of the target group that generates revenue can be smaller than the total target group (Nokia 2003e); this relation is illustrated in Figure 13. For a mobile operator, the total target group is the number of customers.
5.8 Data Transfer per Game Session

A typical game session with The Right Way includes 60 requests to the game server. In a single-player game, the transferred data has been measured to 30 kilobytes. A multiplayer game can reach 40 kilobytes. Most of the transferred data is not payload since The Right Way sends less than three kilobytes of payload in a single-player game with 60 requests. One source of the overhead is packet headers, for example, the HTTP-headers. The amount of data transferred during each request increases with the number of players and the number of requests increase with the play time. The current GPRS-fee is 0.02-0.04 Swedish kronor (TeliaSonera 2004c), which make the fee for a game session 0.6-1.6 Swedish krona (0.1 euro).

5.9 Game Development Projects

Developing a game for a mobile terminal is a smaller project than to develop a game for another platform (International 2003, Nokia 2003f). Only a small team is required to provide the necessary competence: a producer/designer, an artist and a programmer (International 2003). Typically 3-5 people are needed compared to 12-30 for a conventional console game and the development cycle takes a few months (Nokia 2003f). In the development of The Right Way, two people have undertaken multiple roles and completed a multiplayer game in two months time.
6 Development of a Multiplayer Game

This chapter describes the steps in the development process of a multiplayer game. The first topic is the generation of game ideas and then the choice of programming language is discussed. Later in the chapter, the game features are described and motivated and latency measurements are accounted for.

6.1 Game Ideas

It was decided that a multiplayer game should be implemented to demonstrate the capabilities of a mobile network and a mobile terminal. After the decision was made, the first step was to generate a few game ideas. The ideas were evaluated in cooperation with the supervisors at TeliaSonera, to decide which one to implement.

6.1.1 The Right Way (implemented)

A dynamic number of players can participate in this labyrinth game. There are two ways to win the game, by the fastest time or by traversing the shortest path to the exit.

A labyrinth is seen from above but it is covered in darkness. The local player moves a character with a torch that light up a limited area around the player (a light trace has been added). This limited visibility makes it hard to choose the right way towards the exit since there are blind alleys. The labyrinth is displayed in Figure 14. On the way to the exit several contents can be found which adds to the feeling of exploration, for example, a hidden passage behind a bookshelf (replaced by a visualisation of the exit) or the torch goes out when the player falls in a pool of water (simplified) or sweets that make the player go faster (replaced by a time delay). The content icons are displayed in Figure 15.

The local player is continuously displayed in the labyrinth but the opponents can also be displayed (implemented as continuously displayed). Displaying the opponents enhances the feeling of a multiplayer game. The opponents can be positioned closer to the exit or further away, but without seeing the exit it is difficult to tell.
The labyrinths can be exchanged automatically for each game session (not implemented in the demonstration version). The players can participate in the creation of labyrinths if appropriate labyrinth editing tools are made available (not implemented).

![Figure 14: Three players in a labyrinth.]

![Figure 15: Labyrinth contents: start, exit, light, delay and blackout.]

6.1.2 Guess Who

A dynamic number of players can participate in this game. A competition can take place either individually or in teams. The main idea is that an image is presented partially until the whole image is shown. Other graphical filters could be used as well, for example, to blur an image and sharpen it in steps. The players have the
possibility to guess what the image represents by selecting one of the given alternatives. The earlier a correct guess is placed the more points are awarded to the player or the team. The images can be categorized, for example, in celebrities and sports, and fetched from a central database. However, celebrities may not approve that their images is used in the game. Another alternative is to let the players take photographs with the built in camera found in many mobile terminals and send them to a group of friends that take a guess.

A good feature of this game is that it uses images, which are commonly associated with mobile terminals by the introduction of services like MMS and enhanced by the introduction of colour displays. The downside is that few terminals are able to use the camera or access stored images from a Java-application. However, C++-applications have this support.

6.1.3 Four in a Row

This classic game is suited for two players. At the start, both players have 21 coins in separate colours. The players drop a coin by turn in one of seven slots to form piles. The game is won when coins in the same colour is aligned in a row of four horizontally, vertically or diagonally. The classic game rules can be extended to make it funnier to play. For example, some positions in the piles could contain hidden effects that are activated when a coin is dropped in that position. Examples of effects are: to interchange the player’s views, to turn all the piles up side down and to let a player place two coins in a row.

This game was not implemented since it is intended for two players only. Furthermore, it is turn based which do not test real-time capabilities since the players expect a delay while the opponent thinks a move through.

6.2 Choice of Programming Language

Two programming languages intended for games are available for mobile terminals: Java and C++. To decide which one to use, several factors have been examined. The factors are: access to external resources, documentation, components ready to use, spread of terminals, execution speed, software development kits and ease of use.

C++-applications in Symbian OS have the ability to access resources external to the application. Java applications do not have this possibility unless supported by optional packages. In this project, the targeted application does not need to use external resources; otherwise C++ would have been the only option.

Concerning documentation, both programming languages have detailed documentation available on their web sites (Sun 2003, Symbian 2003d). Both languages have many components ready to use, examples are: text fields, lists and image containers.
Today, the number of terminal models supporting Java is greater than the ones supporting C++ (Sun 2004, Symbian 2003c). The spread of terminals is a factor of big importance since it decides the size of the target group. The Symbian OS supports both C++ and Java and it is backed up by many influential companies which make it a good choice for the future.

According to information on the Symbian web site (Symbian 2003d), C++ executes faster than Java. The reason is that C++ is a compiled language whereas Java runs in a virtual machine. Two Java-games have been played, *Speed Devils* and *Siberian Strike* by Gameloft (2003), and the performance was perceived to be good, and would pose no restriction on the targeted game.

Both Symbian and Sun offer free software development kits (SDKs). A SDK makes development for mobile terminals easier since it includes an emulator where an application can be run without having to transfer it to a physical terminal. Judging from documentation and source code examples, Java is easier to get started with. Thus, in a short project more code can be produced in the same amount of time. Based on these factors, the game in this project will be implemented in the programming language of Java.

The choice of programming language on the server side was also between C++ (CGI) and Java (Servlets). The only requirement on the programming language was to be able to receive, handle and respond to HTTP-requests. Since both CGI and Servlets are capable of doing that, the decision follows the choice of the client side (Java).

Two Java-technologies are available to implement a game server; Servlets and Java Server Pages (JSP). However, JSP is more suited for publishing a web site since it is a technology intended to combine static and dynamic HTML. Moreover, JSP are automatically converted to Servlets behind the scenes. Servlets are modules of Java code. The Servlet replies on incoming requests and can use the full Java API. The communication between the game server and the game client does not contain any HTML; hence the server will be written as a Servlet.

### 6.3 The Targeted Terminal

A specific terminal was targeted for the demonstration version of The Right Way: the Nokia 7650, with firmware version 4.46, which is displayed in Figure 16. It is a mobile phone with a large colour display (176x208 pixels) and generous amounts of memory available for games. It also supports a Bluetooth-link, which was used during testing to transfer the game files from a personal computer. Two identical Nokia 7650 terminals were used for multiplayer tests and an additional terminal was used for UMTS-latency tests, a Nokia 6650 with firmware version 3.06.
6.4 Using the Game with Other Terminals

The Right Way uses the full display size of the Nokia 7650 terminal to display the labyrinth. To be able to use the full display size on this terminal, the optional package named Nokia UI has to be used. The display of the Nokia 7650 terminal is 176 pixels wide and 208 pixels high. Without the optional package the same width can be used but only about 70 percent of the height (144 pixels), which is displayed in Figure 17. However, there is only one explicit dependency of the optional package, the class `FullCanvas` in the Nokia UI is used instead of `Canvas` in standard MIDP. Transparent images are also used. However, the transparency support is automatic if it exists, otherwise no transparency is visible. Thus, the use of the optional package can be omitted.

![Nokia 7650 terminal](image)

Figure 16: A Nokia 7650 terminal.

To be able to play The Right Way on another terminal, the size of a labyrinth has to be taken into account. The labyrinth source files, which are described in chapter 7.1.4, present no limitation on the display size. However, the graphics used to build the labyrinth are currently pre-made of a specific size. To suit other display sizes the graphics can be redrawn by an artist or be generated dynamically by the game. An example of dynamic behaviour is to construct a wall by using bricks; the number of bricks needed to finish the wall depends on the wall size, which in turn depends on the display size.

The Right Way uses game actions to control the player in the labyrinth. As said in chapter 3.2.1.4, game actions are guaranteed to function in an appropriate way on any terminal. The last factor concerning portability is memory use. A
local single-player session use 350 kilobytes of execution memory as reported by a call to `totalMemory` in the class `Runtime`. This is generally not a concern for Series 60 terminals, for example, the Nokia 7650 allows up to 1.4 MB of execution memory to be shared by all running applications (Nokia 2003k). However, many Nokia Series 40 terminals have only 200 KB of execution memory, which is not enough to run The Right Way in its current state. The Right Way requires 60 kilobytes of storage memory. The Nokia 7650 excels in the amount of available storage memory with 4 MB and Series 40 terminals generally have 64 KB.

Series 60 terminals fulfil the current memory requirements of The Right Way and Series 40 terminals should not be excluded at this stage since The Right Way has not gone through a memory optimisation procedure, for example, five kilobytes of storage memory can be saved by placing the character images on the server.

The Right Way has been successfully installed, run and played on Nokia 6650, a Series 40 terminal. However, it was not possible to transfer the game files to the terminal and then install them, which was the procedure used with a Nokia 7650. This limitation has been verified on Nokia 6650 with firmware version 3.06 and on Nokia 7600 with firmware version 3.01. Instead, the installation had to be performed with the Nokia PC-suite software on a personal computer, with the terminal linked to the computer during the installation.

The use of game actions for controls worked fine when The Right Way was run on a Nokia 6650 terminal. The controls of this terminal were different than for Nokia 7650, the numerical keys 2-4-6-8 was used instead of the joystick. The automatically added exit command found in every menu on the Nokia 7650 terminal was not found on the Series 40 terminal named Nokia 6650.

Figure 17: Different utilization of the screen size.
6.5 Development Tools

The game client was implemented on a personal computer running the Windows 2000 operating system. The tool used was an Integrated Development Environment (IDE) named Sun ONE Studio 4 update 1 Mobile Edition. To provide full functionality, including text editing, compilation, testing the game with a terminal emulator and creation of game-installation files, the following software was integrated with the IDE:

- Java 2 Platform, Standard Edition, version 1.4.2
- Series 60 MIDP Concept SDK Beta 0.3.1, Nokia edition
- Nokia Developer’s Suit, version 2.0

The Servlet was developed on a personal computer running the Linux Debian operating system, version 3.0. The software used during development was:

- Kate text editor, version 2.1
- Java SDK, version 1.4.2_02
- Tomcat Servlet container, version 4.1.29

6.6 Included Multiplayer Features

At the game of Chess web site (Network 2003) they tell people to be patient and not expect to find an opponent right away. A lack of opponents can occur initially or if the game does not become popular. However, if many people start playing, this problem is reduced. A server can be used to bring players together, enable them to chat and receive help (Gamespy 2003a). However, it can still be hard to find opponents at some times, for example, in the middle of the night. If it is possible to play even when not connected to a network a player develop a higher affinity for the game (International 2003). Therefore, it is a good idea to allow lone players to start a game since a single-player session is better than no session at all. To enable offline play, labyrinth layouts have to be generated in a random way at the client to avoid reuse of a labyrinth.

A matchmaking procedure takes place before a game session is started; its components are accounted for in Figure 29. A server is used to bring players together. There is currently no restriction set on the number of players that can participate in a game session. This lack of a restriction also allows a lone player to play if no opponent is found.

It is possible to leave a game before its moderator has started it. A player that leave an already started game is not removed or replaced but appears inactive. It is also possible for a player to pause the game while the other players play on.

When a game is started, the opponents are displayed at their last known position in the labyrinth. When the positions are known it is possible to choose
another path than an opponent at a crossroad. The opponents positions is loaded at best effort, which means that the local player is not directly affected by the other players other than seeing them, thus a long delay/disconnection does not hinder the local player from continuing to play.

The results can be compared at the end of a game session. If some players have not finished it is possible to watch them progress towards the exit.

6.7 Motivations of Game Features

This section describes game features in detail. Alternative solutions are discussed when appropriate and a motivation for the chosen alternative is given.

6.7.1 Easy for Beginners

The number of in-game actions is kept to a minimum, only directional movement is used. This simplicity makes it possible to enjoy The Right Way even the first time it is played. It is also possible for the players to decide their next move by scanning the screen (International 2003). This is a good practice since casual players do not read a manual before playing (International 2003). To help the players understand the rules, instructive texts are displayed before a game session starts and when content is found in a labyrinth. These instructive texts further diminish the need to read a manual before starting to play.

To test more than one type of skill, as discussed in chapter 2.3, two different measurements are used in The Right Way. The use of blind alleys also adds an amount of luck.

The in-game user interface is kept simple and intuitive by the use of game actions described in chapter 3.2.1.4. One can argue that it would be more flexible to let the player select which keys to use. However, the terminal manufacturers have the expertise to choose buttons that suit each terminal model.

6.7.2 Short Game Sessions

A typical game session is short, a minute or two, to allow a quick game, refer to chapter 2.1. However, it is possible to create longer game sessions by combining labyrinths as discussed in chapter 7.5. If a player has to make a pause in a multiplayer game, it is not a good idea to pause the game for the other players if it is possible for them to play on. Instead one can use artificial intelligence (AI) to fill the position. However, it is hard to mimic a real player and the presence of an AI player makes a game less fun (Gamespy 2003b). If a pause is long or even permanent, it is better to allow another player to replace the one who paused (Gamespy 2003b). A third alternative is the one implemented; a player is not
directly affected by the opponents’ actions even though they are constantly present.

6.7.3 Customized Players

To conform to the ideas in chapter 2.2 players can be customised by their name and character image. It is currently allowed for two or more players to choose the same name and character. However, the players cannot be mixed up behind the scene since they are currently given a unique identification number on a per session basis. Nevertheless, if an online leader board is introduced, then permanent registrations with unique player names are necessary to guaranty that the players can identify themselves on the leader board.

6.7.4 Background Light

The background light of the display is turned off after 16 seconds of inactivity in the Nokia 7650. Without the background light the display is almost unreadable, especially in dusk conditions. A button must be pressed to turn the light on again since it is not possible to control the background light from within a Midlet in a Series 60 terminal, for example, the Nokia 7650 (Nokia 2003i).

6.7.5 No Sounds

Sounds are not used in The Right Way. This exclusion conforms to the ideas presented in chapter 3.3. It has not been confirmed whether or not Midlet sounds is supported by the Nokia 7650 terminal. However, several games are supposed to have sounds but they have not been audible, for example, Siberian Strike (Gameloft 2003).

6.7.6 Feedback when a Button is pressed

In The Right Way, the player is given instant feedback when a valid button is pressed (Nokia 2002c). The only exception is in the labyrinth when the player tries to go into a wall. This lack of feedback can make the player confused since nothing happens when he or she wants to move, even if it is through a wall. A solution can be to temporarily enlarge the wall or display a no-entrance sign. The cause of this problem is related to the tiny walls, which is discussed in chapter 6.9.
6.7.7 Standard Component Menus

There are a number of alternative ways to display a menu. One theory is that the appearance of menus should conform to the general style of the game (Nokia 2003j). In Snake EX (embedded in Nokia 7650) a standard-component pop-up menu is used. When the menu is not visible a colourful image is displayed instead. In the menu of Siberian Strike (Gameloft 2003) only one menu item is shown at a time. The items are traversed sequentially. Like Snake EX a colourful image is displayed. A third and forth approach is to constantly display all elements with or without a colourful image in the background.

The menus in The Right Way are made of standard components (implicit lists). The menus are displayed in Figure 29. From a developers point of view it is a time consuming task to reinvent the wheel. In this case, ready-made components are available for creating menus. A player is likely to be familiar with navigating in the terminals’ embedded applications, which use standard components. Therefore, it helps if the menu system of a game looks and behaves in the same way as other applications, even if a chance to show a colourful image is missed.

6.7.8 Use of a Game Server

If game clients communicate directly with each other, without passing through a server, the latency measured in chapter 6.8 can be reduced. The Session Initiation Protocol (SIP) can be used to create connections either peer-to-peer or peer-to-server. Although SIP-specifications are available for Java the support is not yet found in the terminals. In the mean time an intermediate game server is needed for peer-to-peer communication.

When using a server, it is also possible to relieve the pressure on the terminals. If a game includes heavy calculations, the server could do the work and then return the result. The server could also compile statistics and create top lists for the users’ amusement - to keep them interested in playing. Large databases with statistics could be stored on the server to let the users search the information that they are interested in.

6.7.9 Placement of Game States

If a game has several states located at the clients and an event occurs, for example, when a key is pressed; it has to be transferred one-way to the other terminals to keep all game states updated (Cronin 2001). If the network latency is high, the remote event will be received later than those generated locally, thus the game states can become unsynchronised (Neuenhofen 2003). A single game state can also be used; it can be located either at one of the clients or at a separate server.
A single game state is not the solution to all problems. In the presence of high latency, all events first have to reach the server host; the game state is updated and the change is transferred back to the clients. Several seconds can pass from the time when a key is pressed to the time when the effect is shown on the display (Counter-Strike 2003). This implies that the player whose client has the least latency to the server gets an advantage in the game since he or she can act faster than the other players. In general, the latency is reduced if the game server is located near the clients (Cronin 2001).

A server or the clients can store the current game data when a game is paused. It is possible for a turn-based game to play with seconds, hours or even days between the turns. In general, all types of games benefit from the possibility to pause, for example, to receive a voice call (Nokia 2003f).

The chosen implementation uses several game states located at the clients. The in-game effects of a delay are minimal since the players do not affect each other directly. Instead the information about the opponents is used in a best effort way. When a new position update arrives it overwrites the previous one.

6.7.10 HTTP Connection Type

A HTTP 1.1-connection is used to communicate with the game server. This connection type is the only one whose support is mandatory for all MIDP 1.0 compliant terminals (Nokia 2002a). Therefore, in respect of the connection type, the game is portable to any other terminal. However, there is also a downside with HTTP-connections. Server-initiated communication is not possible; the client has to poll the server to receive new information, for example, the list of available games.

Two types of request methods are available with HTTP-connections, get and post. The get method is commonly used for requesting static contents, for example, web pages (Nokia 2002a). All request information is included in a Uniform Resource Locator (URL). The get method is commonly used in web browsers.

Implementations of the get method can limit length of the URL. Therefore, the post method is more suitable when sending large amounts of data. In addition, the post method allows data to be sent in other formats than the URL.

The Right Way has been successfully tested with both request methods. However, the final choice is the post method.

6.7.11 The Server is polled for Updates

The game client, implemented in MIDP 1.0, polls the server to check if new information has arrived from the opponents. This principle is bad practice since the client has to check for new information even though none are available. The
introduction of the push registry in MIDP 2.0 solves this problem by enabling a server to act on its own as soon new information is available. The push registry also reduces the average latency since the information can be sent without having to await a poll from the client.

6.7.12 A Wait Screen during Network Access

If the network service becomes unavailable for a player in a multiplayer game, it is important to handle the situation gracefully (Nokia 2003j). As discussed in chapter 2.1, the other players should be able to keep on playing. Another point is that the local application should not lock the user interface when a connection is lost. Instead, the user should be given the option to abort a request that is perceived to be too time consuming (Knudsen 2002a). Accordingly, it is appropriate to inform a player of the ongoing communication by displaying a wait screen (Knudsen 2002a). The actual length of the delay is not critical. A more important factor is how the delay is perceived (Nokia 2002c).

A good way to hide a delay, for example, during a period of lengthy calculation or network access, is to display a wait screen (Nokia 2002c). A wait screen also shows that work is in progress, which prevents a player from trying to start the same request multiple times (Knudsen 2002a). Another trick is to move the timely task out of the system thread, to keep the user interface responsive and allow the user to cancel (Knudsen 2002a). A game should react quickly, in a visible or audible way, when the player presses a key. Otherwise, the game will feel unresponsive (Nokia 2002c). To indicate that the game is still alive and processing a request an animated indicator should be displayed (Nokia 2002c).

In The Right Way, network accesses are performed by a separate thread. If there are any problems during the network access, the player has the option to abort the request. A wait screen is used to hide network accesses and to inform the player of the request status. The wait screen contains an animation of two blinking eyes; the animation indicates to the user that work is in progress. The wait screen is illustrated in Figure 29. The wait screen is only used for first time requests, for example, to receive the list of available game sessions. For subsequent requests, the old list can be displayed instead of the wait screen. This reuse is logical when the list is continuously updated while still displayed. When a player returns to the list from a game session room it is possible to display the wait screen while requesting an updated list. However, since an old version of the list is cached, the old list is displayed and may still be correct. This method let the player go back from a game room to the list of available games without having to wait. If the old list should be outdated, an update arrives in a few seconds.

When displaying the results table at the end of a game there is no need to display a wait screen since information about all the participating players are stored locally. A request for the opponents’ results is sent immediately when the local player finishes. However, only the local player is guaranteed to have
finished when first displaying the results screen, which means that the opponents’ results may not yet be available.

Only the network accesses are put in a separate thread. When the game is started the wait screen is displayed while all images are loaded from persistent storage. The next lengthy processing occurs when a game is created. To hide this delay a game is created in the background before it is actually needed. The downside with this procedure is that two game objects exist simultaneously, which consumes more runtime memory. Finally, informational alerts are used to hide any processing done when the player finds content in the labyrinth, for example, when the labyrinth is blacked out.

6.8 Latency Measurements

To see how mobile networks perform, two types of latency measurements have been performed. The first type shows the round-trip time and the second type shows the in-game request time.

6.8.1 Round-Trip Time

A performance measurement of data networks is the round-trip time (RTT); a packet of data is sent from one host to another and then back again. The duration of the transfer is the round-trip time.
The first measurement was performed by Hägglund (2004). The targeted link was between a Sony Ericsson T610-terminal and a Domain Name System (DNS) server in the GPRS-network. The DNS-server is connected to the GPRS IP-backbone, which is shown in Figure 18. The areas involved in this measurement are shadowed in Figure 18. The T610-terminal was connected with a USB-cable to a personal computer (PC). Ping commands were executed at the PC and the amount of data sent was 32 bytes. The measured round-trip time varied between 750 and 1050 milliseconds over a few sets.

According to Lundqvist (2004), the allocation of a time slot in the radio access network takes about 800 milliseconds. Furthermore, an inactivity of 200 milliseconds or more, depending on the settings used by the network operator, can result in the release of reserved time slots. Thus, the round-trip times regarding the DNS-server include one time slot allocation and the time between each ping command exceeded the inactivity limit of 200 milliseconds. Tests with several packets sent within 200 milliseconds were not performed since packets sent that frequently are not used in The Right Way.

The second measurement was performed by Lundqvist (2004). This time the targeted link was between a wireless application protocol (WAP) gateway and the game server. The WAP-gateway is located outside the GGSN and is connected to the Internet, which is where the game server resides. The area involved in this measurement is the External Network in Figure 18. Again, the amount of data sent was 32 bytes. The RTT was stable at 27 milliseconds (Lundqvist 2004). Thus, more time is spent in the mobile network than on the Internet.
6.8.2 In-Game Request Time

To see how The Right Way performs in multiplayer mode, the in-game request time (IGRT) has been measured with both a GPRS- and a UMTS-terminal. The IGRT is measured as the duration for transferring a Hypertext Transfer Protocol (HTTP) request from the game client to the game server, for the server to process the request and for the HTTP-response to be transferred back to the game client. For each request a time stamp is set in the game client before connecting to the server and another stamp is set when the response is received from the game server. The difference between the time stamps shows the IGRT. The transport layer protocols available for communication with the game server in the Nokia 7650 terminal are WAP and TCP. The WAP-protocol can be used between the terminal and the WAP-gateway and the TCP-protocol is used between the WAP-gateway and the game server. An alternative is to use the TCP-protocol all the way. The WAP-protocol is similar to TCP but it is intended for communication between mobile terminals and the Internet. Both protocols use a handshake process to establish a connection. This process results in a delay since the packet exchanged in the first round-trip does not contain any game data. The minimum values for the in-game request time, which can be seen in Figure 21 and Figure 25, hints that the handshake process used by TCP and WAP adds one time slot allocation delay to the round-trip time in chapter 6.8.1.

For TCP, a persistent connection can be used to avoid the handshake process for all but the first packet. Measurements for persistent connections have not been performed with The Right Way; instead a new TCP- or WAP-connection is set up for each packet sent. Measurements performed by Nokia (2003) shows that persistent connections for TCP improves the latency by one time slot allocation delay compared to non persistent connections, which is reasonable.

A commonly used transport protocol for real-time traffic is the User Datagram Protocol (UDP). This protocol is suitable for sending the player position updates in The Right Way. Unlike TCP and WAP, UDP does not include a mechanism for resending a lost packet, which adds a delay. In The Right Way, it is not critical if a packet containing a position update is lost since the position updates are treated in a best-effort way and a new position update is sent the next time the player makes a move. An informal measurement showed that packet loss occurs in GPRS; however, it has not been investigated to what extent packet loss occurs. Even if UDP would be used for position updates, TCP can still be suitable during match-up, to ensure that all players are acknowledged before starting a game session. The UDP was not available for the terminal targeted for the demonstration version of The Right Way.

During the latency measurements, the game server was run on a personal computer, which was connected to the Internet via an Internet Service Provider (ISP) named Com Hem. The bit-rates are specified by the ISP as up to 2000 kilobit per second from the Internet to the server, and up to 400 kilobit per second from the server to the Internet. For each measurement session the minimum-, the
maximum- and the average in-game request time was read from the terminal. Thus, the graphs include several minimum-, maximum- and average values, one per session.

For GPRS, separate measurements were made for WAP and TCP. When using WAP, twelve measurement sessions was performed, with a total of 788 in-game requests. An overview of the results can be seen in Figure 20. The most frequent results were 2500 milliseconds for the average value, as can be seen in Figure 23, and 1600 milliseconds for the minimum value, see Figure 21. The maximum value varies from four to 27 seconds as can be seen in Figure 22. Refer to Table 4 in Appendix B: for more details. Compared to measurements by Nokia (2003l), the values for The Right Way are about one time slot allocation delay higher. It has to be investigated further what causes this difference and how to recreate this lower latency. Nokia have a minimum value that is below 1000 milliseconds, which would be preferable to use with The Right Way instead of 1600 milliseconds. Measurements for TCP using non persistent connections gave similar results as those obtained by Nokia (2003l). 18 measurement sessions was performed, with a total of 1166 in-game requests. An overview of the results can be seen in Figure 24. The minimum value was 1900 milliseconds, which can be seen in Figure 25, and Figure 27 shows an average value around 3 seconds. The maximum values are displayed in Figure 26. The majority of the maximum values are higher for TCP than for WAP. More details on the values from the TCP-measurements can be found in Table 3 in Appendix B:. In total, WAP has slightly shorter delays than TCP according to the measured values in Figure 20-Figure 27.

The Right Way has also been tested with an UMTS-terminal. The test consisted of three sessions with a total of 1000 in-game requests; refer to Table 5 in Appendix B: for more details. The result was an average in-game request time of 800 milliseconds, less than half the times obtained for GPRS. The average values were close to the minimum values and both the average- and minimum values were stable during the three sessions, as can be seen in Figure 19.

A delay of more than one second for an action game (Nokia 2004) makes it hard to respond to the actions of an opponent since an event is already outdated when it is received. Thus, a long delay makes multiplayer action games a bad experience (Nokia 2003f). However, the effect of latency on a game varies with the game type and the skill of the player (Nokia 2004). The priority for mobile networks is currently to maximise the bandwidth, not to minimise the latency (Nokia 2003f). This approach is feasible for non real-time applications that can accept a delay.

While playing The Right Way a latency of two or three seconds is acceptable. A player typically moves four steps in the labyrinth during three seconds, which visibly makes the opponents take long steps in the labyrinth when a position update is received. Latency less than one second makes the opponents move visibly at the same rate as the local player. A latency of eight seconds can allow the opponents to move from one side of the screen to another in one go, which is not acceptable.
Some game types minimise the effects of network latency. For example, gameplay can be local and afterwards the results are sent over a mobile network to be compared with the opponents’ results (Nokia 2003f). In a turn-based game with a round-robin scheme players expect a delay while the other players make their moves (Nokia 2003f). However, if the number of players is large, then multiple latencies and multiple delays are added and each player faces a long delay, for example, 24 players that add ten seconds each give a delay of four minutes. Consequently, the round-robin scheme is not suitable when the number of players is high. Another approach is to require only a periodical login, to give instructions for autonomous game play (Nokia 2003f). An example is a virtual pet that needs to be fed at regular intervals.

Figure 19: Distribution of minimum-, maximum- and average latency for UMTS.
Chapter 6: Development of a Multiplayer Game

Figure 20: Distribution of minimum-, maximum- and average latency for GPRS using WAP.

Figure 21: Distribution of minimum latency for GPRS using WAP.
Figure 22: Distribution of maximum latency for GPRS using WAP.

Figure 23: Distribution of average latency for GPRS using WAP.
Figure 24: Distribution of minimum-, maximum- and average latency for GPRS using TCP.

Figure 25: Distribution of minimum latency for GPRS using TCP.
Chapter 6: Development of a Multiplayer Game

Figure 26: Distribution of maximum latency for GPRS using TCP.

Figure 27: Distribution of average latency for GPRS using TCP.
6.9 Problems during Development

Adding commands to lists, for example, implicit lists, made the Java list implementation in the Nokia 7650 terminal add an exit command at the left soft button. As a result, the exit command was removed from the main menu to avoid having two ways to exit from the same screen. However, according to chapter 6.4 the additional exit command was not present in the Nokia 6650 terminal, which then displayed no exit command at all. Furthermore, commands can automatically be replaced by a synonym that is decided more suitable by the terminal implementation. All commands that are added automatically use the language setting of the terminal, which can clash with the in-game language. For example, if the language of the game is English and the terminal setting is Swedish. A list of choices alternates between the two languages.

For selecting one and only one element in a list when choosing a character image the appropriate component, which is the exclusive list did not behave in a logical way. Two types of visible selections were available, one used for traversing the list and another to select an element for usage. The demonstration subjects interpreted the traversing selection as the selection for usage. Thus, their choice of character was discarded. To remove the danger of misinterpretation an implicit list was used instead.

If no games are found in the available games list then the misleading text “No programs” is displayed instead. This phenomenon is part of the Java list implementation in the Nokia 7650 terminal and may not affect other terminals. However, it is good to notify a user that a list is empty but the above text was confusing.

Since it is not possible to keep the background light on forever as described in chapter 6.7.4 it is important to consider when the game is affected by a lost of sight. In The Right Way there is little chance of inactivity while the local player is moving around the labyrinth. However, when setting up a game or when waiting for an opponent to finish a player is more likely to be inactive. In these situations the only risk is that the player uses the right soft button, which returns to the previous menu and end the game session respectively.

At early informal end-user tests it was noticed that the labyrinth walls was hard to see, as displayed in the left part of Figure 28. This problem originated from the fact that half the wall profile is hidden in the darkness of a parallel path, which has not yet been traversed. As a result, it was difficult to differentiate a wall from a doorway, thus it was not clear where to go next. Two modifications of the graphics have been made to bring out the doorways, as displayed in the right part of Figure 28. The walls have been made thicker, and a trail has been added to the floor, in the style used to mark emergency exits.

Sometimes an error message is displayed when the user exits The Right Way to show that a resource was not released properly before the Midlet was closed.
However, all resources are released by the Java Application Manager when the Midlet is closed so the error message should not imply any problems using the terminal without restarting it. It has not been investigated what cause this infrequent error.

A Symbian specific implementation flaw (Symbian 2004b) required the output stream, which is used to send data, to be flushed explicitly before it was closed. The symptoms are that the client randomly stops sending data. However, whether adding the flush-command solved the problem or not has to be investigated further.

Figure 28: The visibility of labyrinth walls was enhanced in response to feedback from end-user tests.

When trying to download The Right Way over-the-air from a regular web server the result was somewhat confusing; after the download was complete the user is notified that the JAD- and JAR-files does not match. However, the error message was due to the lack of MIME-type specifications at the server. When the URL to one of the game files was entered in a web browser on a PC, the server displayed the contents of the file as plain text. When entering a URL to the game server, that has the correct MIME-type settings, the response was different. Instead of displaying the contents of the file, the user was asked what to do with a file of the correct type, as specified by the MIME-types. Consequently, is not possible to download a game from a server that lacks support for the Midlet MIME-types. 0 provides details about the correct MIME-type settings.

The Tomcat Servlet container caused some minor problems during the game server installation, which is described in Appendix A:A.2. The documentation was somewhat insufficient concerning how to get Servlets to execute on the server. In Tomcat’s manager it is possible to add web-applications, but that requires the directory structure to be already created. When the directories are
created and the Servlet classes have been copied there, the rights must be set to make the Servlet executable. The rights can be set individually for every Servlet in the configuration file of a Servlet, or globally for all Servlets on the server. During the development phase, it is a good idea to use the global configuration file. When using the global configuration file, unnecessary modification of rights is avoided if a file is added. The following code snippet makes it possible to execute all Servlets (the star acts as a wildcard) and it is taken from the global configuration file \tomcat\conf\web.xml:

```xml
<Servlet-mapping>
  <Servlet-name>invoker</Servlet-name>
  <url-pattern>/Servlet/*</url-pattern>
</Servlet-mapping>
```
Chapter 7: System Description

7 System Description

In this chapter the game system is described. The client- and server side are presented separately displaying state charts and class diagrams. The client and server are tied together by the specification of the client-server communication.

7.1 Client

The game development phase began with an adjustment of the chosen game idea to make it more specific about what features should be implemented and how. Objects were identified from The Right Way game idea in chapter 6.1.1, for example, a labyrinth and a player. The object properties and methods were also identified to make class specifications. Menus and other visible screens were sketched on paper along with the image files needed.

The game client implementation began with the construction of basic classes to be able to use them to construct other objects, for example, several blocks form a labyrinth. Each class was tested separately to eliminate errors before adding them to the system of classes. Standard building blocks were also used; they are specified in an application programming interface for Java, which are included in the Software Development Kit (SDK) listed in chapter 6.5.

When a labyrinth was displayed in the terminal, a player was added and button presses was caught to move it. After the functionality of the labyrinth screen was completed, the menu system was created.

7.1.1 Menu System

To allow a user to customize its appearance and other tasks a menu system is used to navigate in the game. The menu system is displayed in Figure 29.

7.1.2 Class Diagram

The game client is constructed of own made classes as well as ready made standard classes. The relations between all classes are displayed in Figure 30. The variables and methods in the own made classes are displayed in Figure 31.
Figure 29: Client menu system.
Figure 30: Client class relations.
Figure 31: Client class contents.
7.1.3 Game Walk-Through

When The Right Way is started a wait screen is displayed for a few seconds while the game initializes. All screens are displayed in Figure 29. The wait screen also appears during match making. After the initiation is complete the main menu is presented with three options, four options when a game session is paused. One of the options leads to a brief help text and another to information about the game and its authors. More important are the options to start a new game and to continue a paused game.

When creating a new game the first step is to enter a name and select an image to differentiate the local player from its opponents. The next step is to check if there are any game sessions available to join. In the list of available game sessions, the number of participants is displayed after the name of each session. If a suitable game session is found it is possible join it. Otherwise, the local player can create a new session for others to join. When a game session has been created or joined the currently participating players are displayed. To reflect changes, the lists of available game sessions and participating players are automatically updated.

The player who created a game session decides when to start the game. All players have the possibility to leave a game session at any time, for example, if the game is not started in a reasonable amount of time. When a game is started all players are positioned at the starting point of a labyrinth. From the beginning, the labyrinth is blacked out except for the starting point. Spotlights are turned on when the local player moves to a previously unexplored area as displayed in Figure 14. The game idea of The Right Way is to find the exit of the labyrinth faster than the opponents or by traversing a shorter path. The local player controls its character by using the joystick on the Nokia 7650 terminal. Other terminals may use different key mappings.

On the way to the exit, different contents can be discovered; they are shown in Figure 15. The light content spotlights the exit, which is positive. The delay content adds a few seconds to the total time of a player, which is negative. Finally, the blackout content turns off the spotlight at any previously visited locations, which is negative.

When the local player finds the exit, his or her results are displayed for a few seconds. The player can then keep looking at the other players as they progress towards the exit or display a results table. It is possible to go back and forth between the labyrinth and the results table. When a player finds the exit, the player is no longer displayed in the labyrinth and its results are sent to the opponents. When the local player feels content with the game, for example, when all players have finished, he or she can go back to the main menu.
7.1.4 Labyrinth Construction

A labyrinth consists of square blocks located in a grid pattern. Every block has a wall combination and content. A labyrinth is displayed as a large image but it is constructed by combining small generic images according to a layout description, which consists of the following fields separated by semicolons: labyrinth name; width in blocks; height in blocks; [Wall combination; Content;]. The fields in brackets can be duplicated.

The wall combinations have integer identifiers as follows where \( N \) = wall to the north, \( W \) = wall to the west, \( E \) = wall to the east, \( S \) = wall to the south, \( n \) = no wall: \( nnnn \) = 1, \( Nnnn \) = 2, \( nWnn \) = 3, \( nnEn \) = 4, \( nnnS \) = 5, \( NWnn \) = 6, \( NnEn \) = 7, \( NnnS \) = 8, \( nWEn \) = 9, \( nWnS \) = 10, \( nnES \) = 11, \( NWEn \) = 12, \( NWnS \) = 13, \( NnES \) = 14, \( nWES \) = 15, \( NWES \) = 16. The contents have the following identifiers: \( \text{empty} = 0, \text{start} = 1, \text{exit} = 2, \text{blackout} = 3, \text{delay} = 4, \text{light} = 5. \)

7.2 Server

The game server listens for requests from the game clients. When a request is received it is processed and an answer is returned.

7.2.1 State Chart

The Servlet has only one state; it listens for incoming HTTP-requests. When a request arrives, it is examined on condition that the shared data in the server is available for use. The shared data consists of game and user information. If another request is being processed, the shared data is not available for use and the incoming request is queued, as the loop shows in Figure 32. The request is processed as soon as the shared data is available for use. This lock synchronizes the access to the shared data, which prevents one request from writing to the same data as another request. If several requests were allowed to change the same data at the same time, the data would become inconsistent – the outcome of the wring is unknown.

When the shared data is available for use, the packet is examined; the data fields is extracted and placed into a vector for easy management. The first element of the vector is the packet type information; the packet id. The remainder of the vector is sent along to a method corresponding to the packet id to generate a reply. There are a total of six types of replies to nine types of requests.

The method called uses the data in the vector to, for example, create a game or update a player’s position in the labyrinth. The reply to the client is based on the received data and it is formatted according to the packet specification in chapter 7.3. For a more detailed description of the Servlet structure, refer to the class diagram in chapter 7.2.2.
Figure 32: The Servlet flow diagram showing the only state and all the actions.
7.2.2 Class Diagram

The game server is constructed of own made classes as well as ready made standard classes. The relations between all classes are displayed in Figure 33. The variables and methods in the own made classes are displayed in Figure 34.

Figure 33: Server class relations.
Figure 34: Server class contents.
7.3 Client-Server Communication

When The Right Way was working properly for one player, the server and client was connected by the addition of a communications module. A packet specification was created to make the client and server speak the same language. The specification tells what data should be sent and the order of the data inside packets.

Every packet has a packet type identification number, which enables both the client and the server to determine what to do with a received packet. The packet type identification number range from 101 to 109 when packets are sent from the client and from 201 to 206 when packets are sent from the server. For most packets, more data values follow the packet type identification number. The data values are divided by a punctuation mark. The packet specification deals with a total of 15 packet types.

To get a clear picture of how a packet is assembled and what it contains, Figure 35 shows an example. The client has at this point already received a list of available games to join. A request is sent from the client containing information to join the game session with identification number 2.

<table>
<thead>
<tr>
<th>Packet ID</th>
<th>Character</th>
<th>Game Session ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>104</td>
<td>6</td>
<td>2</td>
<td>Edward</td>
</tr>
</tbody>
</table>

Figure 35: A packet sent from the client with a request to join a game.

The server receives the packet shown in Figure 35 and establishes that the packet type identification number conform to a join game packet. The next field tells what character the player wants to represent him or her in the game. The game session identification number follows, informing the server what game the player wants to join. The last data field in the packet contains the name of the player. All the above data is registered on the server and a response is generated and sent to the client. A response can look like the example in Figure 36.

<table>
<thead>
<tr>
<th>Packet ID</th>
<th>Player ID</th>
<th>Game State</th>
<th>Player ID</th>
<th>Character</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>206</td>
<td>3</td>
<td>10</td>
<td>1</td>
<td>7</td>
<td>Chris</td>
</tr>
</tbody>
</table>

Figure 36: A packet sent from the server in reply to a request to join a game.

The client receives the packet in Figure 36, examines the packet type identification number and establishes that it is a join game reply. The next field is the player’s identification number for this game session. This number is used to identify the player in all of the following requests concerning the current game.
session. The state of the game session is represented by the number in the next field. A game can have three different states: created, active and finished (10-12). The game state field is followed by three fields that are shadowed, which means that they can be repeated in a sequence depending on the number of players in the chosen game. The information in the three fields is the player’s identification number, character and name. When the client receives this information, it stores all the new player information in a list of players. All packets are described in Figure 37 where PID means packet type identification number and the second row of each packet is the data type used in the implementation. Shadowed data fields may be repeated.

Figure 37: Packet specifications.

### Game Creation

**Client**: Create a game.

<table>
<thead>
<tr>
<th>PID 101</th>
<th>Character</th>
<th>Name</th>
<th>Game Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>String</td>
<td>String</td>
</tr>
</tbody>
</table>

**Server**: Game created; send the player’s new identification number.

<table>
<thead>
<tr>
<th>PID 201</th>
<th>Player ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
</tr>
</tbody>
</table>

### Update Position

**Client**: Send position.

<table>
<thead>
<tr>
<th>PID 102</th>
<th>Player ID</th>
<th>Pos X</th>
<th>Pos Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>int</td>
<td>int</td>
</tr>
</tbody>
</table>

**Server**: Send opponents’ positions.

<table>
<thead>
<tr>
<th>PID 202</th>
<th>Player ID</th>
<th>Player Pos X</th>
<th>Player Pos Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>int</td>
<td>int</td>
</tr>
</tbody>
</table>

### Game List Request

**Client**: Request the game session list.

<table>
<thead>
<tr>
<th>PID 103</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
</tr>
</tbody>
</table>

**Server**: Send all available game names, number of players and the game session identification numbers.

<table>
<thead>
<tr>
<th>PID 203</th>
<th>Game Session ID</th>
<th>Number of Players</th>
<th>Game Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>int</td>
<td>String</td>
</tr>
</tbody>
</table>
Chapter 7: System Description

Join a Game

Client: Join the specified game.

<table>
<thead>
<tr>
<th>PID 104</th>
<th>Character</th>
<th>Game Session ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>int</td>
<td>String</td>
</tr>
</tbody>
</table>

Server: Send the player’s new identification number, game status and all player information.

<table>
<thead>
<tr>
<th>PID 206</th>
<th>Player ID</th>
<th>Game Status</th>
<th>Player ID</th>
<th>Player Character</th>
<th>Player Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>int</td>
<td>int</td>
<td>int</td>
<td>String</td>
</tr>
</tbody>
</table>

Game Status

Client: Request game status.

<table>
<thead>
<tr>
<th>PID 105</th>
<th>Player ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
</tr>
</tbody>
</table>

Server: Send game status and all the player information.

<table>
<thead>
<tr>
<th>PID 205</th>
<th>Game Status</th>
<th>Player ID</th>
<th>Player Character</th>
<th>Player Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>int</td>
<td>int</td>
<td>String</td>
</tr>
</tbody>
</table>

Game Finished

Client: Game finished; send results.

<table>
<thead>
<tr>
<th>PID 106</th>
<th>Player ID</th>
<th>Time</th>
<th>Number of Moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>long</td>
<td>int</td>
</tr>
</tbody>
</table>

Server: Results received; send result table.

<table>
<thead>
<tr>
<th>PID 204</th>
<th>Player ID</th>
<th>Player Time</th>
<th>Player Moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>long</td>
<td>int</td>
</tr>
</tbody>
</table>

Results Table

Client: Request results table.

<table>
<thead>
<tr>
<th>PID 107</th>
<th>Player ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
</tr>
</tbody>
</table>

Server: Send results table.

*See PID 204.*

Starting a Game
Client: Start the game.

<table>
<thead>
<tr>
<th>PID 108</th>
<th>Player ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
</tr>
</tbody>
</table>

Server: Send game status and all player information.

*See PID 205.*

### Cancel / Delete Game

Client: Cancel / Delete the current game.

<table>
<thead>
<tr>
<th>PID 109</th>
<th>Player ID</th>
<th>Creator (t/f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>char</td>
</tr>
</tbody>
</table>

Server: Send all available game names, number of players and the game session identification numbers.

*See PID 203.*

#### 7.4 Limitations of the Demonstration Version

Only one labyrinth is available and it is delivered with the game client. The labyrinths should be stored on the game server and downloaded to each player at the start of a game session. This would also make it easier to add new courses since they are not tied to the game client. The server could also store all the available character images, which would save five kilobytes of storage memory in the client. A newly added character would thereby become directly accessible for download by all users.

The game client only uses one concurrent request to the game server. Thus, if a new request is ready to be sent and the response from the previous request has not yet been received, the new request is queued until the response from the previous request has been received.

Information about using the demonstration version of *The Right Way* with another terminal than the Nokia 7650 is found in chapter 6.4.

The server will not detect a player-drop out from a game session that has been started (reached the labyrinth screen). If player-drop outs were detected the players would be informed that they do not need to wait for a dropped out player to finished. Furthermore, the server could also benefit from this since it does not remove a game session until it is certain that all the players have finished playing.

The player’s name has to be retyped for every restart of the Midlet. This retyping can be avoided by saving the name in the record management system (RMS). Another approach is to store the name on the server and return it when the client identifies itself with a unique id or a cookie.
If excluding the start and exit, there are only three different contents for the players to experience. The number of contents should be increased to create a higher variation of the events in the game.

The list of results presented at the end of a game is not sorted. The list should be sorted on the player names, the play time or the number of moves, depending on the choice of the player.

7.5 Ideas that Can Improve the Game

Players could be able to affect each other directly during a game session, for example, a mine could be picked up and then dropped in another place. When an opponent hits it, the health of the player character could decrease. The players could for example, put all the objects into a case and then open and close the case to see all picked up objects. The choice of object to drop could also be handled from the case. The downside with a case is that the player will not be able to see the labyrinth at the same time due to the small display size and it might also complicate the usage of keys. However, there is also another less complicated way to affect the opponents directly. Instead of picking an object up, it could be activated instantly, for example, the labyrinths of the opponents could be blacked out.

Adding a registration procedure could prevent players from appearing under the same names. This is useful for example, if leader boards are introduced.

To complement the manual matching of players an automatic matching could be implemented. In that way a game could be started as soon as two or more players are available, thus reducing the number of stages and key presses needed before a game session is started, which is recommended by Tsang (2002).

The shared data on the server side is now organized so that read- and write calls cannot be performed at the same time. The data could be reorganized to allow concurrent read- and write calls from several game sessions. This reorganization would result in a shorter queue of requests and thereby reduce the delay of the reply in case of high traffic.

A labyrinth editor can be provided to allow players to create their own labyrinths. However, part of the game idea is that the labyrinths are unknown to the players a priori. Thus, it is not straightforward to allow manual selection or textual identification of labyrinths since a player who is good at memorizing labyrinth layouts can get benefits.

The labyrinths could be classified by difficulty. This would allow a creator of a game session to select the degree of difficulty. A labyrinth could also be larger than the display or a number of labyrinths could be played in a sequence. The winner is the one who walked the shortest way or completed the labyrinths in the shortest time, depending on the choice of game type.

On the server there could be different wall clothing (skins), to get variation in the appearance of a labyrinth. When new wall clothing is added to the server, the
clients get direct access to it and can compare it to their previous favourite skin. A fixed number of wall clothing could be cached locally, depending on their size, to allow a quick change.

The amount of transmitted and received data could be registered for every latest game. If the player has specified the price per kilobyte in the settings of the game, the total price for every latest game could be shown.

Different localizations of The Right Way could be available on the server for download - to offer the players a game in their native language. An alternative is to include a number of languages in the game from the start, and if an additional language is needed, it could be included in the next release of the game.

If the network at some point is not reachable or a player just wants to play a single-player game, an official single-player alternative would be of use. The labyrinth could be generated in the client and the competition could lie within finishing the labyrinth in a better and better time. If the player gets bored of a labyrinth, a single click to generate a new labyrinth would do the trick.

The user could be notified if there is a new version of The Right Way available for download. A description of the new features could also be presented to the player.

The characters could be animated when they are moving from one block to another. However, the animation should not be too long since it can make the player experience the game as slow.
8 Conclusions

Games for mobile terminals are already taking part of the huge turnovers in the computer games industry. In September 2003, tens of games were available at gaming portals (Comviq 2003, Gameloft 2003). However, only a handful of them were multiplayer games that utilize mobile networks. The reason is that the current business models do not encourage multiplayer games. A share of the traffic fees generated by multiplayer games needs to cover the cost of hosting game servers.

Compared to single-player games, multiplayer games can take the entertainment to a new level by introducing social and competitive factors. The success of prior handheld game systems and multiplayer games for PCs makes it likely that multiplayer games for mobile terminals will be a success in the future. An end-user poll or a small-scale deployment of multiplayer games can be suitable ways to find out.

In a mobile situation, the recommended way to add a new game to a mobile terminal is to download the game files over-the-air through a mobile network. However, most terminals on the market today use the first of the two available Java-versions, which implies that they are not guaranteed to have this support. When a game has been downloaded, a typical game session takes place whenever there are a few minutes to kill.

Most terminals on the market support games, written in either Java or C++. Both languages have similar capabilities but games written in Java do not have access to resources in the terminal, for example, the address book. Even with this limitation, Java is the preferred language since more terminal models support Java than C++. Java has a core set of functions supported by all Java terminals. If other functions are used, a game needs modification before it can be run on a terminal that only supports the core set of functions. The second version of Java gives some game specific enhancements. However, as of February 12 2004, only four terminals supporting Java MIDP 2.0 is available on the market. Unfortunately, inconsistencies with the Java standard and other software errors are common in the terminals, which can make the appearance of a game vary.

The three types of mobile networks, GSM, GPRS and UMTS, are all capable of transferring in-game data. However, GPRS and UMTS are preferable since network resources are allocated only when a transmission takes place.

During the development of The Right Way no hardware constraints have been experienced. However, other terminals are specified to have more limited amounts of memory. The client- and server part of the game have been tested and are
working satisfactory. However, the communication from the client to the server occasionally fails in the game. The cause of this failure has to be investigated further.

The in-game request time has been measured for GPRS and it is considered too high to be used in real-time games. The cause of the latency is time-slot allocations and the currently available network protocols. However, upgrades in the mobile networks and the addition of the UDP-protocol in the terminals can improve the situation.

The in-game request time for UMTS has also been measured. The results are promising with latency less than half the one obtained for GPRS. This result implies that multiplayer games are more suitable for UMTS than for GPRS. However, turn-based games are still possible for GPRS since a delay is expected while the other players make their moves.

Consequently, it is important to consider the latency when designing multiplayer games for mobile networks. There are numerous methods available to reduce the user-perceived latency. The Right Way uses one method, to accept deliveries of new information in a best effort way. However, other methods could also be used, either as a complement or a replacement, and are therefore suggested for further research. Examples are: to use several concurrent connections to the game server and to interpolate the received information.

If introducing on-line tournaments, security issues have to be taken into account. For example, it is possible to cheat by creating a forged game client that either visualize hidden information to the user or send faked information to the server. Another concern is the possibility to copy a game file and give it to a friend. However, if most of the revenue from a multiplayer game comes from continuous game play, this procedure might not be a problem.
9 References


Counter-Strike review, 2003, Most popular online game, but the best one? 8 April 2003. URL: http://www.dooyoo.co.uk/games/pc_games/half_life_counterstrike/_review/414870/ (11 October 2003)


Nokia Corporation, 2003l, Multi-Player MIDP Game Programming. Version 1.0; 29 October 2003. URL: http://nds1.forum.nokia.com/ndsd/ForumDownloadServlet?id=3838&name=Multi%5FPlayer%5FMIDP%5FGame%5FProgramming%5Fv1%5F0%5Fen%2Epdf (10 February 2004).


Smith, Tony, 2003, Mobile terminal handset sales picking up. 1 August 2003. URL: http://www.theregister.co.uk/content/68/32118.html (30 September 2003)


Sun Microsystems, 2000b, MIDP 1.0 specification. URL: http://www.jcp.org/aboutJava/communityprocess/final/jsr037 (29 September 2003)


TeliaSonera, 2003a, Botfighters Positioning Game. URL: http://botfightes.teliomobile.se/ (20 October 2003)


TeliaSonera, 2004b, Telia innehållstjänster. URL: http://wap.teliomobile.se/ (10 February 2004)

TeliaSonera, 2004c, Teliamobile.se. URL: http://www.teliomobile.se/ (11 February 2004)

Tsang, Eddie, 2002, Elements of Successful Mobile Multi-Player Game. 22 November 2002. URL:


Appendix A: Installation Instructions

This appendix describes the procedure to install The Right Way. Both the client and the server are needed to run The Right Way in multiplayer mode. For single-player sessions only the game client is needed.

A.1 Game Client

The game client consists of two files, a Java Application Descriptor (JAD) and a Java Archive (JAR) that need to be transferred to the terminal. For testing purposes a Bluetooth connection was used. When transferred with Bluetooth to the Nokia 7650, the files arrive in the messaging inbox. To install the game the JAD-file should be opened. It is important to open this file since it contains a description of the Midlet needed by the Java Application Manager (JAM) to install the game correctly. After the installation is complete, The Right Way can be found in the general applications menu /tools/applications. Of course, it would be preferable to find the game in the top-level menu /games where the embedded games are found. A few Nokia terminals support this feature but not the 7650 (Nokia 2003c). The terminal manufacturer decides where to install a Midlet.

For over-the-air (OTA) provisioning of a Midlet the JAD- and JAR-files are placed on a web server, in the same directory (Mahmoud 2002). A terminal needs to find only the JAD-file since it contains a URL to the JAR-file. A link to the JAD-file can be published on a WAP-page or be sent to a client via SMS. When the JAD-file is found the Java Application Manager (JAM) check the attributes inside to see if the terminal is able to run the application (Nokia 2002b). If it is, the JAR-file is downloaded and The Right Way is installed in the menu described above. It is possible to request that the JAM sends a notification back to the server after the installation is complete (Nokia 2002b).

The web server used for publishing the files needs to support files with the JAD- and JAR-extensions. This support can be achieved by registering the corresponding Multipurpose Internet Mail Extensions (MIME) types: JAD-files with “text/vnd.sun.j2me.app-descriptor” and JAR-files with “application/java-archive” (Mahmoud 2002, Nokia 2002b).

The WAP-gateway must allow the game files to pass through. Therefore, it must support the same MIME-types as the web server (Nokia 2002b). There is also a requirement to allow large files through. The Right Way has a file size of
60 kilobytes. If the file is too large it is impossible to download it over-the-air (OTA) since the mobile network operators can limit the file size allowed through their WAP-gateways (Nokia 2003a). TeliaSonera currently have a maximum file size limit of 350 kilobytes (Lundqvist 2004).

A.2 Game Server

To make the server and thereby the Servlet to work, there are some hardware- and software requirements. These requirements are:

- Linux or Microsoft Windows operating system
- 150MB of free space on the hard disk drive
- Tomcat Servlet container version 4.1.29
- an IP-address accessible from the Internet
- Java software development kit (SDK) version 1.2

It is important that the IP-address of the server is accessible from the Internet. Otherwise, a connection to the Servlet cannot be established from the game client.

To be able to execute a Servlet, a Servlet container has to be used. A well-known Servlet container is Tomcat – a reference implementation of the Servlet specification. Tomcat is a part of Apache’s Jakarta Project, a project that “creates and maintains open source solutions on the Java platform for distribution to the public at no charge” (Apache 2004).

When Tomcat is installed, some environment variables have to be set before the server can be started. These variables could for example, be set in the autoexec.bat in Microsoft Windows. The result would look like this:

```
SET TOMCAT_HOME=<path>\tomcat
SET CATALINA_HOME=%TOMCAT_HOME%
```

When these two environment variables are set, Tomcat is started with the start script that has the name startup.bat and is located in the directory \tomcat\bin. The server is shut down with shutdown.bat in the same directory.

To execute a Servlet in Tomcat, a new directory structure has to be created under \tomcat\webapps. The directory structure required is for example: \gameserver\WEB-INF\classes. A file named web.xml has to be created in the WEB-INF directory. This file controls, among other things, the rights that make the Servlet executable. However, the rights were set in a global web.xml, read more about this in chapter 6.9. More information of the contents in web.xml can be found on Apache’s web site (Apache 2004). The Servlet consists of compiled Java-files, which should be located in a directory named classes. To make
Tomcat detect new or updated class-files, the server has to be restarted or the Servlet reloaded.

When connecting to the Servlet, the URL of the HTTP-request must not contain the .class-suffix of the Servlet name. If the directory structure is \gameserver\WEB-INF\classes and the class-file is named GameServer.class, the appropriate URL for a request would be:

http://<servername>:<port>/gameserver/Servlet/GameServer
This appendix account for the data collected during the in-game request time (IGRT) measurements performed with The Right Way.

Table 3: Data from the GPRS latency test using TCP.

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Number of Packets Sent</th>
<th>Minimum IGRT [ms]</th>
<th>Maximum IGRT [ms]</th>
<th>Average IGRT [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>2453</td>
<td>5875</td>
<td>3259</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>2238</td>
<td>3875</td>
<td>2775</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>2406</td>
<td>6781</td>
<td>3417</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>2204</td>
<td>9234</td>
<td>3169</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>2312</td>
<td>11390</td>
<td>3270</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>2109</td>
<td>9016</td>
<td>3031</td>
</tr>
<tr>
<td>7</td>
<td>32</td>
<td>2328</td>
<td>11015</td>
<td>3363</td>
</tr>
<tr>
<td>8</td>
<td>33</td>
<td>2313</td>
<td>9360</td>
<td>3376</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>2172</td>
<td>12203</td>
<td>3461</td>
</tr>
<tr>
<td>10</td>
<td>45</td>
<td>2016</td>
<td>11188</td>
<td>3100</td>
</tr>
<tr>
<td>11</td>
<td>48</td>
<td>2343</td>
<td>9719</td>
<td>3559</td>
</tr>
<tr>
<td>12</td>
<td>63</td>
<td>2187</td>
<td>17031</td>
<td>3420</td>
</tr>
<tr>
<td>13</td>
<td>66</td>
<td>2156</td>
<td>15688</td>
<td>3349</td>
</tr>
<tr>
<td>14</td>
<td>67</td>
<td>2000</td>
<td>10641</td>
<td>2886</td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1969</td>
<td>10360</td>
<td>2653</td>
</tr>
<tr>
<td>16</td>
<td>137</td>
<td>2047</td>
<td>10172</td>
<td>2916</td>
</tr>
<tr>
<td>17</td>
<td>162</td>
<td>2219</td>
<td>27719</td>
<td>3657</td>
</tr>
<tr>
<td>18</td>
<td>260</td>
<td>1953</td>
<td>14843</td>
<td>2965</td>
</tr>
</tbody>
</table>
### Table 4: Data from the GPRS latency test using WAP.

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Number of Packets Sent</th>
<th>Minimum IGRT [ms]</th>
<th>Maximum IGRT [ms]</th>
<th>Average IGRT [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>1688</td>
<td>6828</td>
<td>3607</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>2093</td>
<td>7250</td>
<td>2684</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>1703</td>
<td>7062</td>
<td>2526</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
<td>1625</td>
<td>7297</td>
<td>2503</td>
</tr>
<tr>
<td>5</td>
<td>55</td>
<td>1968</td>
<td>18125</td>
<td>6264</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
<td>1703</td>
<td>4171</td>
<td>2490</td>
</tr>
<tr>
<td>7</td>
<td>62</td>
<td>1593</td>
<td>7594</td>
<td>2386</td>
</tr>
<tr>
<td>8</td>
<td>72</td>
<td>1657</td>
<td>7454</td>
<td>2580</td>
</tr>
<tr>
<td>9</td>
<td>92</td>
<td>1610</td>
<td>7547</td>
<td>2435</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>1672</td>
<td>7891</td>
<td>2434</td>
</tr>
<tr>
<td>11</td>
<td>110</td>
<td>1640</td>
<td>27625</td>
<td>3088</td>
</tr>
<tr>
<td>12</td>
<td>138</td>
<td>1843</td>
<td>16141</td>
<td>3206</td>
</tr>
</tbody>
</table>

### Table 5: Data from the UMTS latency test.

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Number of Packets Sent</th>
<th>Minimum IGRT [ms]</th>
<th>Maximum IGRT [ms]</th>
<th>Average IGRT [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>300</td>
<td>745</td>
<td>2703</td>
<td>816</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>726</td>
<td>4112</td>
<td>810</td>
</tr>
<tr>
<td>3</td>
<td>400</td>
<td>717</td>
<td>6402</td>
<td>812</td>
</tr>
</tbody>
</table>