Metropolia University of Applied Sciences
Institute of Technology
Degree Programme in Media Engineering

Compatibility Heuristics for Modern Mobile Games

Paul Gitau

Bachelor's Thesis: 28 October 2009

Instructor: Juho Pakkonen, QA Manager
Supervisor: Petri Vesikivi, Lecturer

Metropolia University of Applied Sciences
Institute of Technology

ABSTRACT

Author	Paul Gitau
Title	Compatibility Heuristics for Modern Mobile games.
Number of Pages	55
Date	28 October 2009
Degree Programme	Media Engineering
Degree	Bachelor of Engineering
Instructor Supervisor	Juho Pakkonen : QA Manager, Digital chocolate LTD. Petri Vesikivi : Lecturer

The purpose of this study was to demonstrate ways to perform a compatibility test for a modern mobile game. These tests were performed by running a tastephone midlet to the mobile handset to determine its characteristics and the equivalent implementation. Compatibility check test was also aimed to help a specified mobile game application to run successfully into a mobile phone.

Compatibility heuristics for modern mobile games was also designed to study the factors that should be considered before integrating mobile games to mobile handsets. These applications are identified to determine optimal reliability and to utilize defect counts to the respective devices. This document also discusses on identifying the criteria for determining compliance of a mobile phone with the specification of the game application.

The test case for this document outlines practices for testing a managed game application or component for compatibility with newer and older mobile handset versions. It covers different configurations you should test to ensure that the application continues to run properly and successfully. It also discusses features of a good mobile game.

The result of the test case was to determine whether a game fully runs as proclaimed to function with a particular mobile component. The results also indicate the game compliance to a mobile phone to uncover flawed hardware and application programming interfaces combinations that decrease the game's ability to function properly.

Keywords	mobile games , J2ME, compatibility testing

Contents

Abbreviations and Terms

J2ME

A free and open Java platform enjoys the status of an industry standard backed by all major handset makers.

Localization

The process of translating a software application to a new language. It involves translating all text to the new language and in some cases changing some of the images or resources.

TasteMidlet

An application downloaded to a mobile handset to determine features and characteristics of a phone.

API

An application programming interface is a source code interface that an operating system, library or service provides to support requests made by an application.

High-Low end handset

Can be recognized by its profile for example high-end handsets have MIDP 2.0 profiles while low-end have MIDP 1.0 profiles.

BREW

Binary Runtime Environment for wireless devices used in application development platform for mobile phones. Brew was created by a company called Qualcomm and its aim was to create and develop a software platform that can download and run small programs for playing games.

TAF

Stands for *Tell a friend* option used in a game for sending sms to friends informing them about the game.

OTA

Stands for *Over the Air*. Software that enhances connectivity between two points. Mainly used to download mobile games to a handset.

MMS

Stands for Multimedia messaging service. It allows sending multimedia messages that includes multimedia objects such as images, audio and video.

SMS

Short Message Service (SMS) is a communication protocol allowing the interchange of short text messages between mobile devices.

UMTS

Stands for Universal Mobile Telecommunications System. UMTS is one of the third-generation (3G) cell phone technologies. Currently, the most common form of UMTS uses W-CDMA as the underlying air interface. It is standardized by the 3GPP, and is the European answer to the ITU IMT-2000 requirements for 3G cellular radio systems.

Java verified program

The Java Verified™ Program provides a streamlined testing process for mobile Java technology applications, enabling developers and wireless operators to confidently develop, deliver and monetize mobile applications.

CLDC

Connected Limited Device Configuration. A J2ME configuration for mobile devices.

API

Application Programming Interface.

JVM

Stands for Java Virtual Machine.

KVM

Stands for Kilobyte Virtual Machine.

GSM

Global System for Mobile Communications.

GPRS

General Packet Radio Service.

1 Introduction

This thesis study focuses on J2ME applications. J2ME is a free and open platform that enjoys the status of an industry standard backed by all major handset makers. It has the ability to work on different handsets which are Java enabled and also run on different operating systems. A J2ME file requires JVM (Java Virtual Machine), also known as a KVM (Kilobyte Virtual Machine) for mobile devices. It also consists of the Mobile Information Device Profile (MIDP) which is API functioned to develop applications for small devices including mobile devices and PDAs. With its latest MIDP version 2.0, it simplifies the game API development process thus reducing the developing period.

For a game developer, the ultimate goal is to create an experience that is fun and challenging for the player with no defects. However, studies shows that one third of mobile games paid for and downloaded by users fail to work owing to compatibility issues. This is revealed by mobile application developers. Defining these compatibility heuristics is a challenge for the mobile entertainment industry, for developers and for the operators who supply the vast majority of games all over the world.

The main aim of mobile games compatibility heuristics is study factors that should be taken into consideration before and after deploying a game application into a mobile handset. Compatibility heuristics enhances solutions to whether an application can run successfully on one or more models of handsets. This is also aimed to verify that the application will function as proclaimed on a wide variety of profiles, toolkits and network configurations. Mobile handsets are not yet standardized, meaning they are unable to produce a consistent usability experience from device to device.

Compatibility testing was done by running a tastephone midlet into several mobile phones to determine their characteristics and features involved in the phones. After the completion of compatibility testing the game application is ready to be deployed to the mobile phone. The final mobile application also has a successful action over various initiatives and functions on different compatible mobile handsets.

2 Background

At the moment, the low-end MIDP 1.0 devices have the largest installed base. However, many new MIDP 2.0 smart phones, including advanced devices that support the J2ME 3D, Bluetooth, and Web services APIs, are starting to take over the high-end market.

A Java application needs to be ported, optimized, and tested for each target device it is intended to run on, which is a complicated challenge. For example, even among Nokia products, Series 60 and Series 40 devices have very different screen sizes, memory sizes, and CPU speeds. Furthermore, the quality control of the Java Runtime Environment (JRE) on devices has been weak. Different devices may have different bugs, particularly in their thread or memory management implementations

Post production was a time-consuming process in this study. During post production, the game application and its handset profile are tested for errors; this helps to keep the development costs low and provides the necessary flexibility with ample support freely available for developers using it.

Testers, engineers and quality assurance personnel are involved in post production and their objective is to analyze, document and rectify defects found on each and every application profile and forward the all the affected ones back to the Game developers . Before a product goes to the post production testing stage, its profile has to undergo a process called compatibility test. Here the application is checked for issues that would enable easy and better communication with the compatibility characteristics.

2.1 Application Development

Compared with other smart client platforms, an application that uses J2ME has unique advantage: it is designed for mobility. J2ME applications run on various devices from numerous vendors that are extremely important in the highly competitive mobile device business due to the large number of different devices. All major smart phone manufacturers have committed to support the J2ME platform. Code portability is crucial for developers who want to maintain a single codebase yet still reach the maximum number of consumers. [1]

The designs of these mobile game applications are always critical. The design process requires the planning and design of a market-viable application. It is inevitable that such a process will involve the creation of novel, valuable and intellectual property. The key to development on any hardware platform is to test early and to test often. [1]

Developers are encouraged to develop prototypes, usually in proof-of-concept and are then required to produce a working mobile application, which is externally tested through a recognized testing program. Conformation to these standards requires a structured approach to testing, and the project management software supports both compatibility testing, bug tracking and formal test documentation. [1]

Java verified program helps with its compatibility procedures to develop, promote and deploy an easier way to use Java related applications. The target for running a java verified program includes contributing to network operators to deploy java applications to their networks and to also help manufactures to provide a valuable service. The ability to work on these projects is constituted by developing programming languages.

2.2 Compatibility Consideration and Specification

This chapter discusses on factors be learned during compatibility checking. These compatibility heuristics for modern mobile games are considered to reduce major challenges in the mobile game industry. Before deployment of a game application to a mobile device, these factors explained below should be taken into account. Most of these considerations are according to the modern mobile games compatibility issues that are backed up in today's mobile games industry.

2.2.1 Platform Variations

Java 2 Platform, Micro Edition (J2ME) is one of the leading application programming interfaces that provide an application environment for mobile devices used by game developers to produce their products, although other platforms such as Symbian and Flashlite are growing fast. Gaming platforms facilitate administration and maintenance of mobile games on mobile handsets and some of these platforms must be able to maintain the following functions; [2]

- Chatting in applications to encourage interactivity between players.
- Interrupted and resumed later in case of a call or low battery.
- Localization capability (supports different languages).
- Applications which are independent of device and have the ability to perform with fewer problems.
- Various billing models (Mostly used in BREW).
- Ability to support for MMS, SMS, Game servers Internet, GSM, GPRS, and UMTS.
- Publishing other games can be inserted in the same applications to enable users to check upcoming games.
- Interfaces (APIs) are made public for game developers.
- Integrations with other users such as sending of text messages to friends informing them about the game and other gaming contents.

Some mobile devices are seen to support only one platform but that is not the case. This is because mobile applications can be written to run natively within an OS or another technology such as Java, which can run within another operating system [2]. Most of the new Nokia handsets which run natively on Symbian platform integrate under no defects with applications that are J2ME enabled. The integration of these platforms can also be examined in a number of ways, for example a US based platform called BREW can run also run Java with no or little downfall.[2]

Platform variations not only depend on the platform techniques, but also on the application domain and device manufactures. Most of this manufactures have their own system of performance and developing their products. It is ideal important for a device of certain platform to undergo a compatibility testing before proceeding to the next phase.

2.2.2 Handset Manufactures

Different platform manufacturers' phones vary a lot, in software development environments, screen sizes and keypads. These fundamentals are mostly focused on mobile gaming. The mobile phones developed by the same platform manufacturer vary as well. The work of the manufactures in game development is to implement platforms without duplicating the protocols to enable a smooth run of the game application. [2]

Nowadays manufactures release different kinds of handset from a low profile (MIDP 1) to a high profile (MIDP 2) all with different kind of features. With the increasing rise of technology, manufactures are finding a better way or launching their handset making it easier for different game platforms to run easily on a mobile.

Mobile phone manufactures are starting to act as mobile carriers. They have their own default menu for the games on the application menu deck inside the phone which acts according to the mobile handset features. They either make by themselves or consult game companies. Most of mobile games must be compatible with mobile handset manufacturers and mobile network operators to enhance their functionality. Every time manufactures develop a new handset, the compatibility tester examines and evaluates its

specifications, and then modifies just one component to develop a compatible version of the game or find a compatible handset for it. [2]

In most cases games that were tested on one handset, can be automatically tested on all compatible handsets without making a different game build for the handsets. This reduces the amount of work and cost. The testing ensures that the final handset can work over different networks and with various handsets, providing the customers with products that really work, and guarantee a faultless user experience.

2.2.3 Handset Model

The functions, features and performance of different handsets may vary. Despite conformation tests and implementation of handsets features, the different makes, platform, soft keys and other features of a handset model can challenge game developers.

With all these challenges coming up, game developers have to put portability of code at the top of the priority list if they are to make successful returns on their investment. Not only do mobile phones differ among different manufactures but also within the same manufactures. The most important aspect to consider is the eligibility of features involved in a handset with the application to be inserted into. Some of the things to be considered when working with the handset models are explained below. [3]

Memory Limitations

When building applications for mobile games, there are three aspects that need to be considered, which are working memory, storage memory and application memory. Working memory is memory that the game is executed during runtime. If the game is too big, an error will occur and the applications will not be executed. Game programmers are now finding better ways to send scores, compare scores, and create user options and other user data in the storage memory.

Application memory is the working memory storage needed to hold all games and applications. Before employing the game into the handset, a compatibility tester is advised to consult the model specifications and see what limit size the phone can hold each game. Many mobile phones also warn the users in case the application memory has exceeded. Although the game can run successful on the handset, it can result to heap thus making the playability slow. The other thing is that the game could crush which might result to starting the application again. [3]

Screen Limitation

Aside from memory another factor to be taken into consideration is the size of the screen for each mobile handset. For example a Sony Ericsson P800 when folded out has pixel display of 208 × 320 and a Nokia 3650 has a display of 176 × 208. A developer has a chance to consider releasing a specific version that includes the appropriate image sizes. Nokia may have sprites the size of 16 × 16 pixels and the P800 may have sprites the size of 32 × 32. Though more and more mobile handsets are being released with color screens, one should take in consideration the millions of existing phones already sold on the market that only have black and white displays. [3]

The variations of J2ME enabled handsets not only differ from range of manufactures, but also from model to model produced by the same manufacturer. Important issues to be considered when adjusting games to the display are screen size, frame rate and color. Game developers have to make a decision, which handsets to support in accordance to the screen size. It is ideal to write the game generically, and then make alternate versions for certain handsets. Once again a game developer should consult the manufacturer’s specification to make the appropriate adjustments.

The figure below illistrates different kinds of mobile phone screen sizes. Developers release applications that match with the screen sizes to enable full visual appearance to the user.The pixels of the three phones are different.

Figure1: Shows different screen sizes from different manufactures.

Depending on the type of content created for a particular handset, game developers can allow changes by creating contents on a size that match the minimum end-screen size, and then allow other content that appear off the screen to be visible on other screen sizes. By doing this the game features will be visible to all the players. [6, 9-11]

The figure below illustrates the contents of the game which are first created on a small screen size and other portions being left for larger screen sizes.

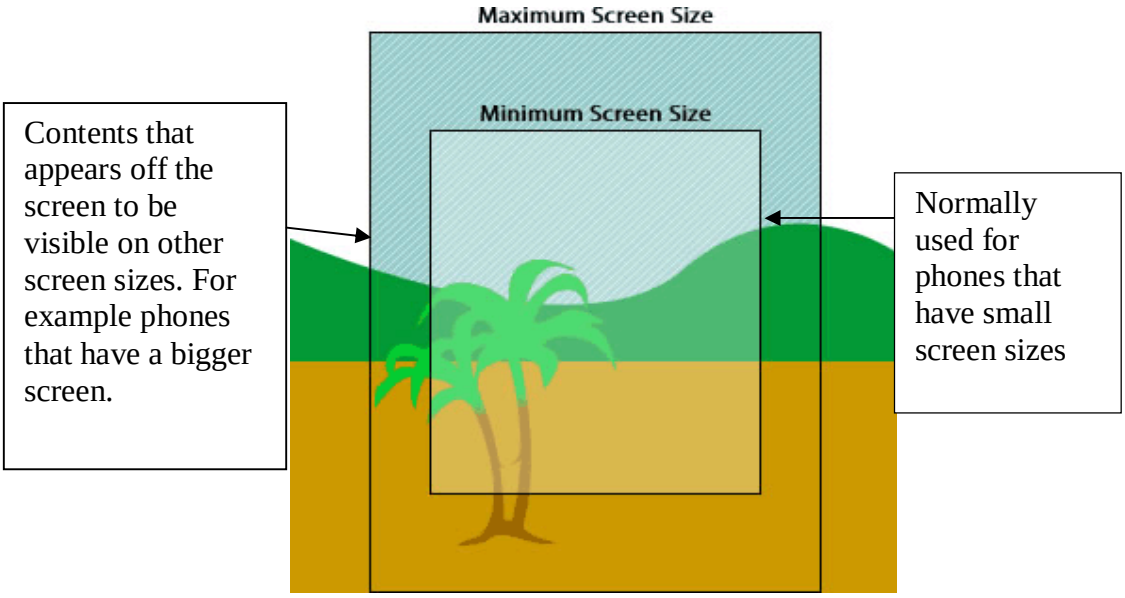

Figure 2: Maximum and minimum screen sizes.
(http://www.adobe.com/devnet/devices/articles/flashlite_multiple_versions_02.html)

Normally the developers would choose to make handsets with a minimum Screen size be the master profile, so that it can also fit to phones that have a large canvas size. [6, 9-11]

Keyboard Variations

Button functional handset and a touch screen phone are the two main challenges that a developer has to consider. It is necessary to make two different builds of code. When making games for a touch screen handset, a game developer might have a different code change because of its way of settings. Most games use buttons to support their game play, but on touch screen the buttons might not be there.

Application Programming Interfaces For Mobile Games

The Mobile Game API provides a consistent Java framework for the mobile industry. It defines an optional package made of a set of Java classes designed to address the problem of creating a wide range of games on mobile devices, with limited resources. Below are some of the APIs in a mobile phone that help with the running of the game.

- Wireless Messaging API supports the sending and receiving of short message service messages via GSM.
- Mobile Media API which provides the ability to play back MIDI, tone, sampled audio, Real Time Streaming Protocol streaming, and music progressive sound files, as well as perform video and image rendering.
- Security and Trust Services API for J2ME including implementations of the Application Protocol Data Unit and other optional packages. These APIs allow game applications to offer high levels of data security, with features such as cryptographic APIs, digital signature service, and user credential management.
- Mobile 3D Graphics API for J2ME provides features to create rich 3D graphics for games, animated messages, custom UIs, and interactive product visualization.
- Wireless Messaging API supports the sending and receiving of SMS and MMS messages.
- Content Handler API allows MIDlet to be specified as the content handlers for one or more specific file types, thereby enabling Java applications to handle multimedia and Web content seamlessly.
- Scalable 2D Vector Graphics API for J2ME enables the rendering of scalable 2D vector images, including external images in Scalable Vector Graphics (SVG) format. The principal uses for this API are in map visualization, scalable icons, and applications that require scalable and rich animated graphics.
- Advanced Multimedia Supplements provide 3D audio and music support that allows applications to provide a rich sound experience for games and multimedia applications. The API is updated to handle audio mixing, including mixing of 3D audio. [15]

2.2.4 Connectivity

Older mobile phones supporting mobile gaming have infrared connectivity for data sharing with other phones. These days the fast and influential growing aspect in mobile gaming communication is the connectivity. Nowadays mobile games are becoming more like a computer. With the increasing production of high end phones, developers are finding better ways to increase interactivity between users. Some of the functions enhanced by connectivity include the following. [4]

- Users can view and receive scores from the server. Normally based in the high scores in the game menu.
- TAF- Gamers can also inform their friends about the game by sending sms or mms. Wireless messaging capability.
- Multiplayer-enhancing communication through connection protocol to share game information.
- GPRS location identification.
- Sending of password and encryption to servers.

Below some of the connectivity used normally for MIDP 2.0 phones will be explained.

Bluetooth

Bluetooth is an open specification that enables short-range wireless connections between two or more mobile devices and thereby simplifies communication and synchronization between devices. A developer must consider only the modern MIDP 2.0 handsets that only have this connection capability. Bluetooth wireless technology uses a globally available frequency band (2.4GHz) and can transmit data up to 2.1 Mbit/s on distances from 10-100 meter. The capability of Bluetooth in Mobile gaming is to connect two short distance gaming interaction and sending games between phones.

Compared to wireless LAN, Bluetooth offers lower bandwidth but is a cheaper technology and consumes little energy. Bluetooth had a slow start but can now be found in most mid- and high-end mobile phones, and is also used in three handheld gaming systems. [4]

Wireless LAN (802.11)

Wireless LAN also called”Wi-Fi”, is a set of standards for wireless local area networks. The 802.11 family contains several protocols such as 802.11b, which features a maximum data rate of 11 Mbit/s, a range of up to 100 meters, and operates in the 2.4 GHz band. [4]

Wireless LAN offers high bandwidth, but consumes a lot of energy. Wireless LAN is today a common technology in connection in mobile games. Wireless LAN is also used in two powerful handheld gaming systems. [4]

Radio Frequency Identification

Radio Frequency Identification is a method of retrieving ID codes over short-range radio. An RFID tag is a small object, such as an adhesive sticker, that can be attached to or incorporated into a product. RFID tags contain antennas to enable them to receive and respond to radio-frequency queries from an RFID transceiver. [4]

Because of its static, passive nature, it is difficult to use RFID as the basis for a multiplayer game. Instead, RFID could be used for incorporating items or passive players into a proximity game. [4]

2.2.5 Localization Variance

Localization is the introduction of a device or service to the legal, linguistic, and technical specifications of a certain location. Localization enables users to interact with software, help, and documentation in their own language, in turn enabling vendors to sell more products and reduce the cost of international support. [5]

Every handset comes with different make and functions. Before porting a game to handset, localization is normally considered into other languages. The five major languages used in localization are EFGSP. English, French, Italian, German, and Spanish are some of the languages that are considered important by carries in their specific countries. With the increasing game complications, a game player might find it difficult to play a game if he does not understand English or other languages good. Many mobile carries in their region accept only localized content. They provide competence and offer better content instead of lowering prices. [5]

The challenge of localization becomes apparent when a required handset does not support a certain language. A developer is forced to code the application according to the required language. This may be cost effective. Certain aspects to be considered when localizing games include: language, character sets numbers, dates and time, and currency.

Language

Framework, menu entries, message boxes such as Tell A Friend, the status bar, and error messages are considered to be important. Most of the build which are downloaded to high end phones have a default language, mostly English. The user can change to other languages in the same application. Most of the low end handsets handle just one application with one default language because of memory limitation.

Character Sets

Every language is considered to have a character set which differ in writing and pronunciation for example English has 128 characters, but that does not apply to other languages. For example:

- Finnish, has ä,ö and so on.
- French accents, has in à, é, î, and ç,
- Spanish punctuation, has the reverse question mark ¿,
- Umlauts, has ß, ä, ö and ü in German or Finnish,
- Other umlauts, has æ and å in Danish, Norwegian or Swedish. [5]

2.2.6 Carrier Specifications.

Currently, mobile games are mainly sold through Network Carriers / Operators portals. This means that there are only a few lines of text and perhaps a screenshot of the game to excite the customer. These network operators have certain specifications, which must be followed by developers and publishers before the final game handover (beta). Some of these specifications include the following [1]:

- Game downloaded can only be directed to the Mobile phone application in the menu.
- Certain play patterns that are recognized by the network operator’s customers.
- Licenses and powerful brands that improve the quality of the games.
- Free demos that will encourage the costumer to purchase the game.
- Interactivity that enables the player to communicate to each other.
- Well-organized subscriber’s carrier billing statement.

The figure below illustrates the sequence of events of a game flow from the developers to the actual users. The flow of the money goes backwards. Each part plays a major role during game development. Also a set of compatibility issues should also be learned by each other to avoid upcoming challenges.

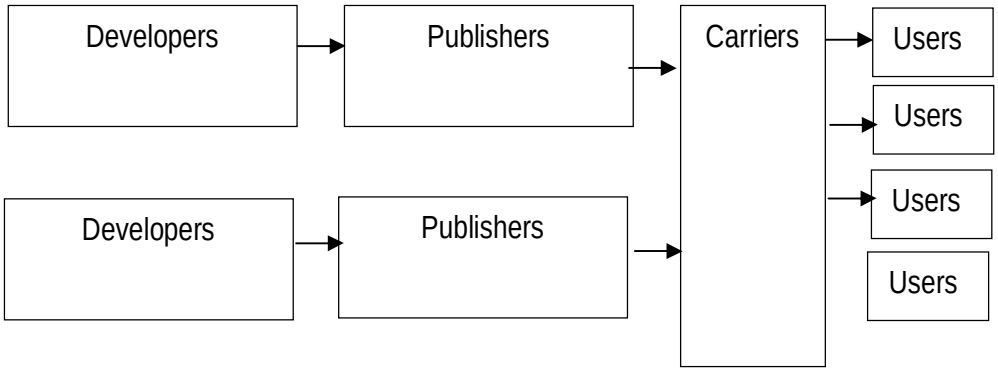

Figure 3: Carrier flow from publishers to users

The consumer is able to download the game only from the carrier deck of the network operator although these days, some are finding better ways to sell their applications through websites.

Carriers are continuing to invest money to upgrading their networks to allow high-speed data. As a consequence more mobile users have access to data enabled handset with higher download speeds which should improve the quality of games. It will be easier and faster to download larger, higher quality games that have more content. An example of a cycle of improving quality driving and being driven by improving technology can be seen in the evolution of the Internet. [1]

2.2.7 Security

The security issues have not had a very big role so far in mobile gaming, although with the increased technology and new mobile knowhow abruptly coming up, there is a growing need to increase security. Mobile carriers are naturally interested in preventing games to be copied to other devices. On the other hand, this is a problem for those who change mobile phones and would like to play a mobile game with their new device.

However, this problem has not necessarily considered a big one since a new phone model would need a new game version anyway. Also, many of the story-oriented current mobile games do not include that much content and are quickly played through. However, as quality of mobile games increases and the games become more connected, the security issues become more and more important.

The combination of connected gaming and increasing popularity of these games will create a need for enhanced security both in a technical sense and when considering cheating and other typical problems in online games. Another issue that will make security even more important is gambling or trading game items for real money. Such games have not yet been seen in the mobile phones, but this may change in the near future. The content will become more important as its value gets bigger.

2.2.8 Sound Formats and Their Variations

A decade ago mobile phones were limited to simple monophonic ringtones. Nowadays the capability of these devices varies depending on a number of factors such as operating system, CPU power and so on. Device manufacturers offer dozens of handset models based on standard and proprietary technologies. There are now a number of differing approaches to playing sounds, including polyphonic MIDI (such as SP-MIDI and General MIDI) and a variety of digital audio recording formats (such as ADPCM and MP3.) One content developer is estimated to offer over 400 different audio implementations on mobile devices. [2]

When implementing games for mobile phones, a developer is advised to confirm the appropriate sound formats of the device in order to design sound that will be compatible with the device. Most of the games developed for low-end handsets that have MIDP 1.0 use monophonic sounds while MIDP 2.0 phones use polyphonic sound.

Formats and platform variations require content providers to maintain multiple tools, and require a substantial investment in product testing. The investment in education alone is a significant barrier to entry, which further restricts market growth. Furthermore, inconsistent implementations and inadequate audio designs increase the

cost of audio production by forcing developers to work harder just to make their content sound as good as possible. It is reasonable to expect that the audio capabilities of mobile devices will increase over time, and additional complexity will follow unless action is taken now to stabilize the market. [6]

Mobile phones manufactures have failed to standardize and open audio formats that can be used by all the developers without striving to develop equivalent audio for the devices. Adopting standardized or open- standard audio formats reduces the need for content developers to produce multiple version of their content. It also facilitates the creative process because of the wide variety of audio production tools available for standardized formats. Other compatibility issues to be considered include battery power and add-ons like joy sticks and integrated digital cameras. [6]

2.3 Device Fragmentation

Several applications are ported to dozens of mobile phones. This creates costs and a lot of time is wasted. The process of producing multiple versions to run on multiple mobile devices is called device fragmentation. As discussed earlier, various initiatives need to be considered before performing this process. Fragmentation drives up the costs and time it takes to get an application to the mass market. Applications have to be ported from one device to another, translated from one language to another, and even customized for specific mobile operator needs. The result is often hundreds of different stock keeping units (SKUs) of the same product. [10]

2.3.1 Causes of Fragmentation

Difference in Platform

As discussed previously, platform variation is one of the major causes of fragmentation. The difference can be seen between Operating Systems such as J2ME, Symbian, Nokia OS, RIM OS, Apple OS X, PalmOS, Mobile Linux, Android, and BREW. API standards like MIDP 1.0 mostly used in older times, and MIDP 2.0 which is the newer version. [10]

Feature Variations

Feature variation differs depending on the mobile device characteristics. For example high end versus low end handsets, light versions versus full versions, canvas size and memory management. Variations and features can be seen on the manufactures websites. Point to note is that the some features might be compatible with other features. [10]

Operating Context

In Operating Context the hardware and software environment of the device, environmental diversity, carrier specifications and the target user are considered. If there are two mobile phones which have the same version using two different mobile carriers, a developer is advised to make two separate builds of code because each carrier uses a different carrier-specific API. [10]

The figure below shows the processes of device fragmentation from the fragmented applications to diversity of users. Softwares have to the fragmented to many applications since different hardwares have different features.

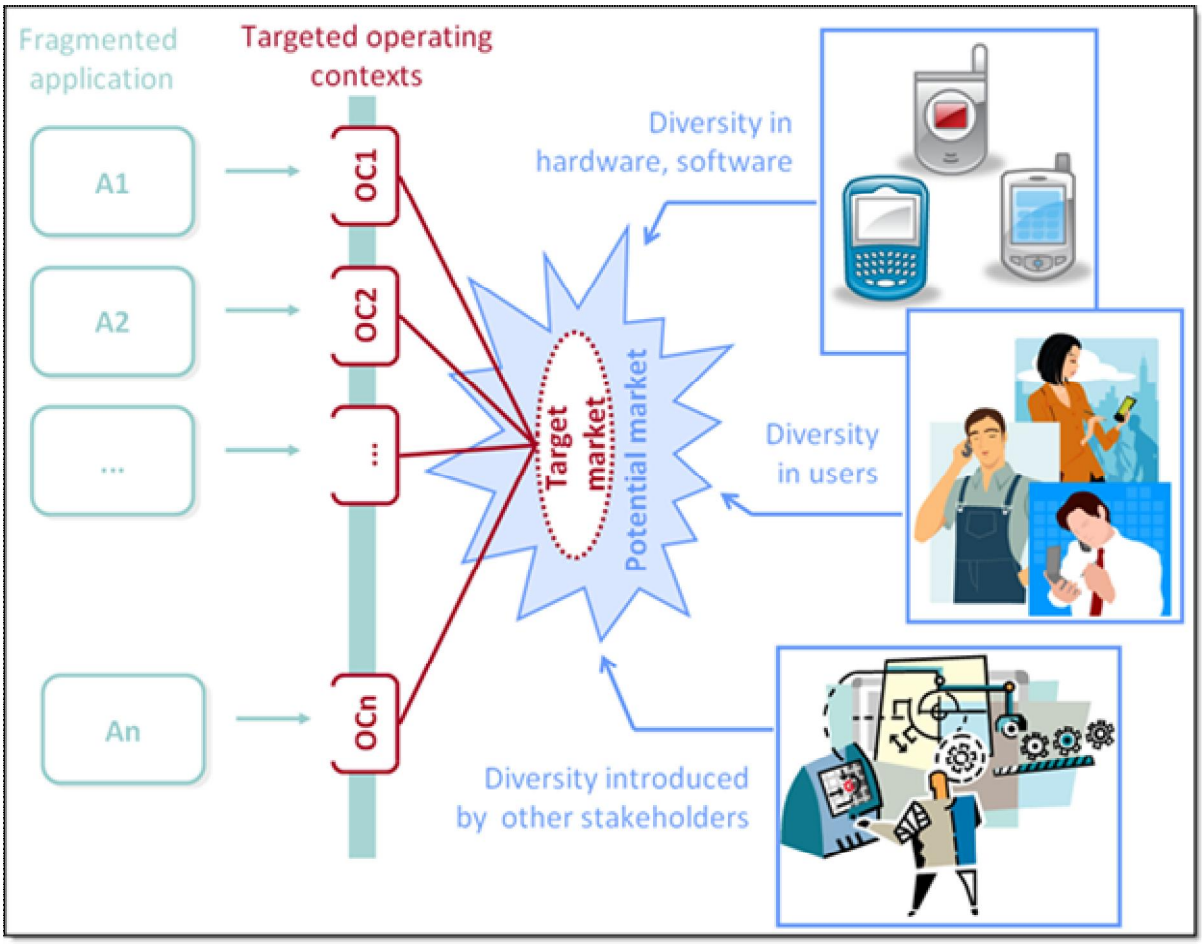

Figure 4: Shows processes involved in device fragmentation http://www.comp.nus.edu.sg/~damithch/df/device-fragmentation.htm. [10]

The figure above also shows the diversity of new device models that are released. Even released applications will continues to fragment, as more and more new operating contexts need to be supported. This fragmentation may be hard to predict, or plan for. It also becomes necessary to identify the operating contexts available before deploying applications to their equivalent devices.

3 Game Characteristics on a Target Device

To build good mobile applications, it is important to understand their key characteristics that define great mobile applications. This chapter discusses how mobile phones incorporate with a mobile game application and what are the most important characteristics of a good mobile game application.

Characteristics of mobile devices will also set some requirements for mobile applications. Users interact with applications by using a standard 12-key keypad and few navigation keys. In some cases they can use a miniature-size joystick, which is operated with the thumb. The small screen size, insufficient audio capabilities, limited processing power, and battery limitations will cause additional requirements that need to be taken into account when designing mobile games. [11]

1.	Audio-visual representation supports the game
2.	Screen layout is efficient and visually pleasing
3.	Device UI and game UI are used for their own purposes
4.	Indicators are visible
5.	The player understands the terminology
6.	Navigation is consistent, logical, and minimalist
7.	Control keys are consistent and follow standard conventions
8.	Game controls are convenient and flexible
9.	The game gives feedback on the player's actions
10.	The player cannot make irreversible errors
11.	The player does not have to memorize things unnecessarily
12.	The game contains help

1 Audio-Visual Representation Supports the Gameplay

Mobile games are increasingly becoming complex with the rapid development of graphic cards, games look visually appealing and the players expect that, too. However, the game graphics should support gameplay and story and be informative for the player. In addition, the graphical look and feel should be consistent throughout the game. Audio can be used to evoke emotions and increase immersion. A good sound environment in the game supports a positive gaming experience. Normally, there are two types of audio present in the game: music and sound effects. Both of these have their own roles in creating the sound environment and they should work together seamlessly and not create cacophony.

The graphics or audio should not prevent the player from performing actions or make them unnecessarily difficult. For instance, using a nice 3D camera effect may look good, but may make the game unnecessarily difficult to play. [11]

2 Screen Layouts are Efficient and Visually Pleasing

Designing a good layout is not always easy. The layout should present all necessary information for the player, but on the other hand, if the screen is filled with all kinds of information, it starts to look crowded. Also, games should follow the general principles of good screen layout design. It is important that the player finds the navigation controls and they should not be mixed with the information that needs to be visible on the screen.

In general, it is important for mobile games due to limited screen space. Designing the layout for a mobile-phone screen can be challenging, but a good rule of thumb is that information that is frequently needed should be visible to the player all the time. [11]

3 Device UI and the Game UI are Used for Their Own Purposes

There is a difference whether the player is dealing with the game user interface or device functions. The game interface should not use the device's user interface widgets in the game interface, because they can interfere with each other. The most impressive interference is achieved when the game uses full-screen mode hiding other features.

In mobile games, some features of the device, for example the network connection, should be visible in the game interface. However, the game should present this information using user interface widgets that are consistent with other elements in the game. [11]

4 Indicators are Visible

The player should see the information that is required for being able to play the game. An example of this kind of information could be the status of a game character. Information that is frequently needed should be visible for the player all the time. The player should always know the current state of the game, for example, whose turn it is to make the next move. Indicators for critical game play information should be presented to the player clearly enough. For instance, the player will feel very frustrated if the game character dies suddenly and there was no indicator that it was starving. However, the "less is more" principle is also important: information that is not critical or used frequently should not be visible all the time.

In mobile devices, there are different indicators for device functions that need to be visible during the mobile game. These can be, for instance, the typing modality of the keypad (such as number, alphabetical, or T9) or the connection indicators (such as Bluetooth). In addition, the player might want to know the battery level and time during a game session. [11]

5 The Player Understands the Terminology

The terminology that is used in the game should be understandable and not misleading or unfamiliar for the players. A similar rule also appears in. Technical jargon should be avoided. For instance, terminology that is related to the game concept or features that the game needs from the device should be translated into more understandable language. In network games, the player usually needs to connect to a game server before the play session can start. The command for doing this could be “Connect to the server.” However, from the player’s point of view, it is not interesting to connect to the server, but rather to join the game. In this case, a more understandable command name would be “Join the game.” [11]

6 Navigation is Consistent, Logical, and Minimalist

The player can navigate in the game menu, which usually consists of settings and selections for the desired game session, or in the game world, if the game has a visual world. In the game UI, different functions should be organized reasonably and possibly on different screens. However, long navigation paths in the game menu should be avoided. Short navigation paths are clearer and easier to remember. In the main game menu, the player should be able to start a game and have access to other important game features.

In the game world, navigation should be intuitive and natural. The game world can be either a 3D world with forests and mountains or a table of cards or another simplified representation in 2D. Regardless of the complexity of the game world, players should be able to navigate there smoothly. With a proper set of control keys, navigation can be intuitive and almost invisible. [11]

The navigation on a mobile device is not easy because of the small screen and limited input devices. Mobile devices have two kinds of navigation controls: permanent and temporary navigation keys. Permanent navigation keys should be used primarily for navigation. Temporary navigation keys are often related to applications or to a specific

user interface style. Since the games do not necessarily need to follow the device's user interface style, the use of these keys can be more flexible. [11]

7 Control Keys are Consistent and Follow Standard Conventions

Using common conventions reduces the time that is needed to learn to use any software application. The same applies to games: using standard control keys reduces the time that is required for learning to play the game since the player can use his or her knowledge from other games. Game devices usually have specific keys for certain actions and every game should follow them.

The mobile device is a relatively new kind of device for playing games, but a few conventions already exist. For example, number five on the mobile device's keypad is usually the selection key. One interesting thing about mobile devices is the design driver of the device. If the device is meant to be operated one-handed, the game should be playable with one hand. Correspondingly, the device's standard input methods should be used for controlling the game. [11]

8 Game Controls are Flexible and Convenient

Novice players usually need only a subset of the controls when they start playing the game. On the other hand, veteran players often need shortcuts and more advanced commands. It should be possible to customize the game controls or use shortcuts or macros. However, using shortcuts should not provide a major edge in a competitive player vs. player game. The configurability and amount of controls needed to play the game should be kept at the minimum, but they need to be sufficient. In addition, the controls should be designed according to the device's capabilities.
Currently, mobile devices may not be as flexible as most of the other game devices and the possibilities to customize game controls are often limited. In mobile phones, some functions are specifically assigned to certain keys and they should be accessible even though the device is used for playing games. This reduces the number of available keys for controlling the game. [11]

The mobile device is primarily used for communication. The player should be able to manage incoming calls during a game session. Two keys are assigned for call handling and they should never be used for controlling the game:

• The Send key (usually identified with a green symbol) is for answering calls. This will also move the game to the background because in-call functions are activated.

• The End key (usually identified with a red symbol) is for rejecting incoming calls.

In addition, some keys are defined dynamically. Following the conventions of using such keys is not always straightforward. [11]

9 The Game Gives Feedback on the Player's Actions

A good user interface has a low response time on the player's actions. An action can be either a single key press or a more complicated input sequence. The player should notice immediately that the game has recognized the action by providing feedback. The most common way of providing feedback is to present it graphically. Other alternatives are to use audio or tactile feedback. Providing only auditory feedback is not acceptable since a player may be playing the game without sounds.

Although the game needs to respond immediately to the player's actions, the consequences of the action can be shown to the player later (see, for instance, the "delayed outcome" game design pattern. If an action cannot be performed immediately, the game should notify the player that it takes time.
In mobile devices, the network connection is usually considerably slower than in other game platforms and it can create latency in the response time, making the game unplayable. Usually the players try to repeat the command because it may seem that the input was not received. Mobile devices also lack processing power compared to other game platforms, but this gap is constantly decreasing. [11]

10 The Player Cannot Make Irreversible Errors

The game should confirm actions that can cause serious and irreversible damage. Also, when mistakes happen, it is helpful to enable recovery. In games, making errors is often part of the gameplay. However, this heuristic deals with errors that are related to the bad usability of the game user interface. Errors often happen when the player deletes game objects, such as avatars or items. Sometimes, however, the errors can be related to positive things that must be done, but the time is just not right. [11]

11 The player Does Not Have to Memorize Things Unnecessarily

The game application should not stress the user's memory unnecessarily. This applies to game user interface design as well. Sometimes, the challenge in the game can be memorizing. [11]

12 The Game Contains Help or Instructions

The game teaches the player what he or she needs to know to start playing the game. The players do not often read manuals, and a mobile game does not usually even have a paper manual. Even though a tutorial mode at the beginning of the game is usually helpful, having a complete tutorial is not well suited for a mobile game. The players do not necessarily encounter or need everything during the first play sessions.
If tutorials are used, they should be entertaining and rewarding, and part of the actual game.

A mobile game should entertain the player even if he or she would only have a couple of minutes available to play it. The player should be able to accomplish something in the game within the first five minutes. Ideally, the tutorial could be embedded completely in the game so that help would be provided every time when it is really needed. Help is also often needed in error situations. If the game provides useful error messages, the player can understand better what caused the problem. [11]

Figure 5 shows a structure of a modern game application.

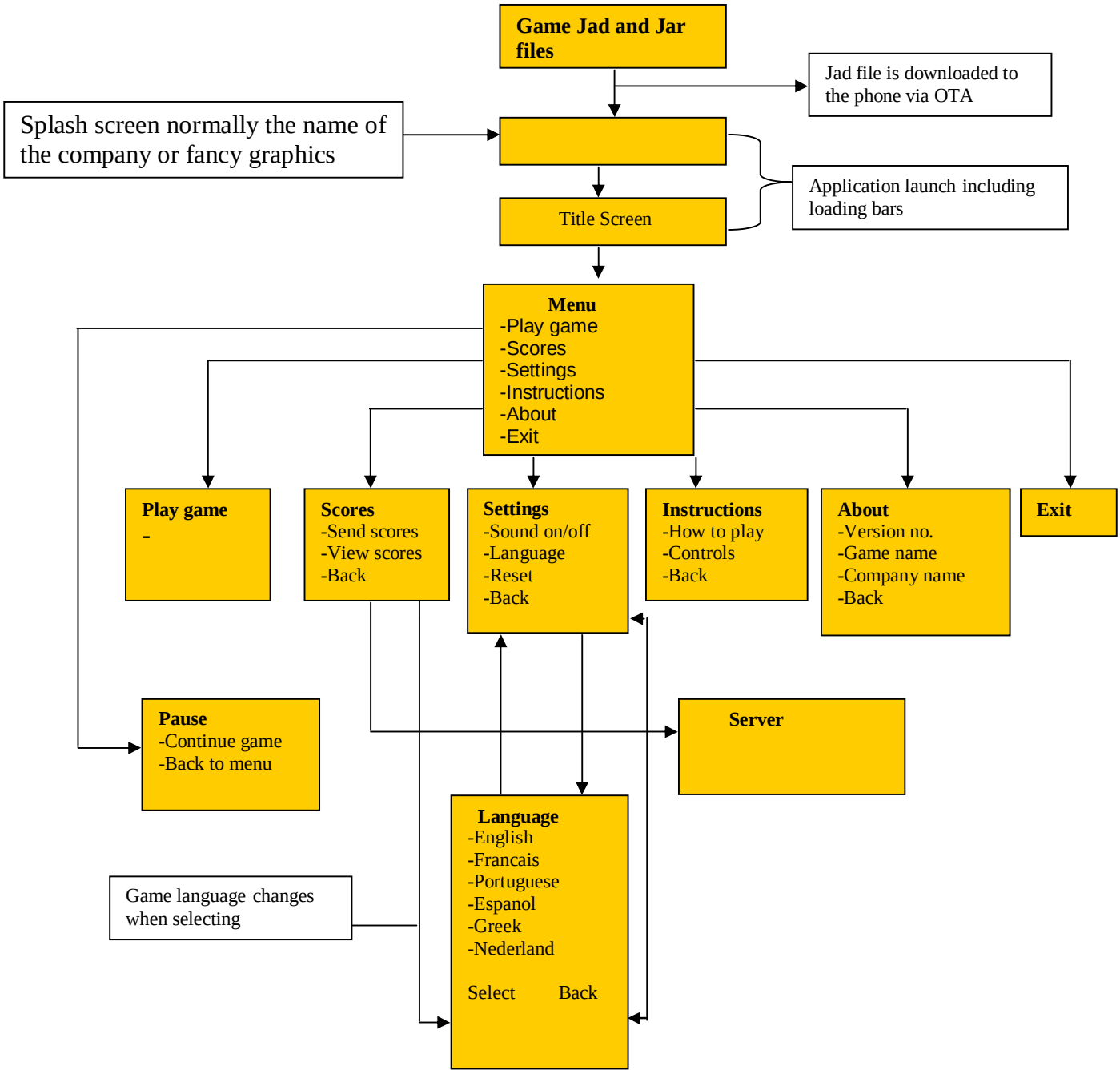

Figure 5: Shows a structure of a modern game.

The figure above illustrates the structure of a modern mobile game mostly downloaded to MIDP 2.0 handsets. Other capabilities such as server communications can also be shown in the figure.

3.1 GameCanvas Size

The main concepts of game canvas differ with phones that have MIDP 1.0 and MIDP 2.0. The developer will have to implement the code depending on the low or high mobile device he or she is working on. The game must scale according to available screen dimensions. If differences are too big then different images must be used for different phones.

In MIDP 1.0 the game developer is not able to take control of the display. The only option is to use the standard sizes of 176 by 144. In MIDP 2.0, the developer can use the GameCanvas to take control of the whole screen. Another option is that the game should use very little MIDP2 functions because MIDP1 phones do not support them. If possible it is better to use only MIDP1. [7]

The process of porting in J2ME is different from the porting for Flash Lite. In J2ME games, developers often hard code every value. The values for moving a sprite, for placing a sprite on a canvas, for setting up a tiled background and for setting up fonts on the canvas during run-time are usually all hard coded. This means that in order to port a J2ME game to a bigger screen size, the hard coded values must be updated. Also, since the graphics in J2ME games are bitmaps, the graphics will need to be redesigned at a larger size and enhanced for a higher resolution screen size. [8,191-235]

Figure 6 illustrates the layers found on a game canvas.

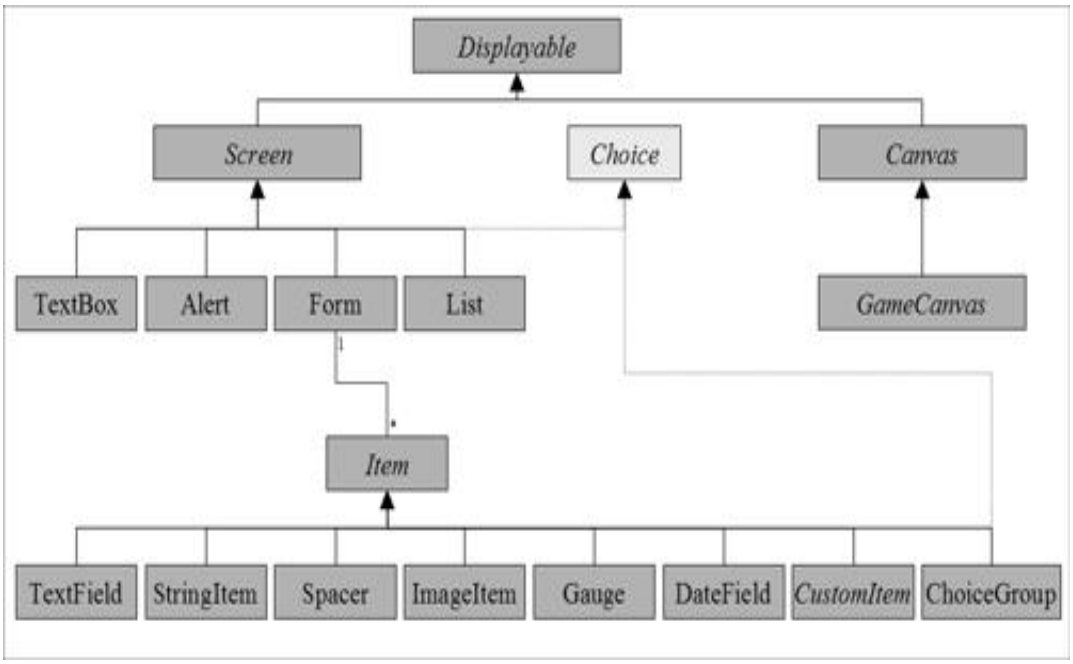

Figure 6: Layers of the game canvas
http://www.actionscript.it/XStandard/Library/J2ME/Fig4_1.png

3.1.2 Layers

A Layer is essentially an object that appears on the mobile screen. Usually they will move, or become animated according to a set of parameters defined by the by the developer for a particular application. [8,191-225]

3.1.3 Layer Manager

The Layer Manager manages a series of Layers. For games that employ several Layers, the Layer Manager simplifies game development by automating the rendering process. It allows the developer to set a view window that represents the user's view of the game. The Layer Manager automatically renders the game's Layers to implement the desired view. [8,220-222]

3.1.4 Sprite

A Sprite is a basic visual element that can be rendered with one of several frames stored in an Image. Different frames can be shown to animate the Sprite. The developer can control which frame is used to render the sprite thus creating animations. The position and movement on the screen is specified by the application, which makes calls to the appropriate Sprite methods. The Sprite class also provides various transformations (flip and rotation) and collision detection methods that simplify the implementation of a game's logic. [8,191-235]

3.1.5 TiledLayer

A TiledLayer is a visual element composed of a grid of cells that can be filled with a set of tile images. This class enables a developer to create large areas of graphical content without the resource usage that a large Image object would require. It is a comprised of a grid of cells, and each cell can display one of several tiles that are provided by a single Image object. [8,213-219]

Figure 7 illustrates variations of the screen sizes in different mobile handsets.

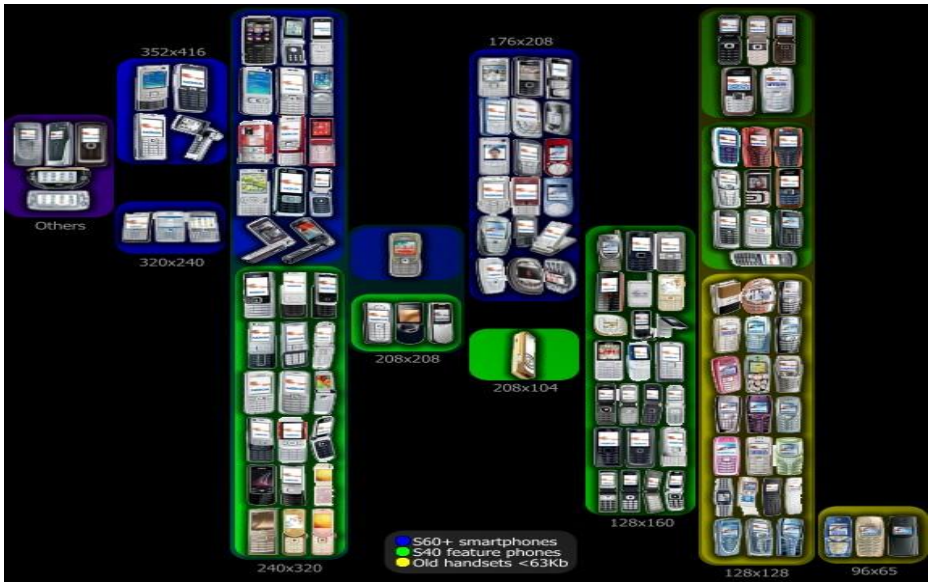

Figure 7:Mobile phones display different screen sizes
http://blog.masabi.com/2008/01/truth-about-mobile-fragmentation.html

3.2 FPS (Frame Per Second)

FPS depends on the performance of the game. FPS rate must get the minimum 8, if the game is too slow to play. Sometimes it is necessary to drop some content such as special effects which needs to be refreshed quickly to increase FPS. Game must be created so that even if FPS changes it should not affect the game.

3.3 Heap Memory

Some devices have limited Heap memory such as RAM. If the heap memory runs out, then the game will crash. A programmer should use memory smartly and sometimes it is necessary to drop content such as big images so that game will not use so much heap memory. Heap memory may sometime affect the FPS since the lesser the content, the faster the game.

3.4 Keypad and Touch Screen

Some phones such as Ngage have eight directional keypad whereas Nokia S60s' have 4 directional keypads. Some phones do not have any such as Siemens SL55. Different versions of game must be made so that game will always support all the available keys. Some devices also have touch-screen and some does not have. A game must support touch-screen if it is available on phone. Some phones also have changeable screen such as Nokia N95. A Game must support both dimensions if the device can change screen size.

3.5 2D and 3D Graphics

Some games have completely different 2D and 3D versions. Depending on the API support in the handset enables the type of graphic to be supported. A game has to have a compatible 3D version of API to display 3D scenes on the mobile device.

4 Handsets Characteristics (Test case)

4.1 Aims of the Project

The aim of this project is to compare two or more emulators' characteristics that are different or have almost identical capabilities. Whether the game application will be able to run on equivalent mobile devices will also be determined. The handset specification has to be tested using a taste application which runs on a java environment. Handsets that are based on different Operating systems can also run on java platform. The taste midlet application is be able to check some features of the handset that are core important to the game. Some of them include Platform compatibility, canvas size, API's, memory run outs, performance Values like 3D, record stores, heap size and sound formats supported. Handset specifications can also be checked in the manufacturer's websites. The comparison between mobile games and mobile phone APIs must be similar in order for smooth running of the application.

4.2 Checking the right devices features.

Select any devices on which you plan to develop mobile games in. Choosing the right devices depends on the game characteristics. Remember that the game will also depend on the provided mobile phone API.

The test case for this project was done by three mobile Nokia phones, one high-end and two low-end phones. By using a taste phone it is possible to check the phone specifications on a java environment. The taste MIDlet jar was downloaded directly to the phone. [12]

4.3 Insert the SIM Card into the Device

Select the operator networks that you plan to develop your mobile games for. The SIM card was inserted into the target device that you have chosen. The operator network was

chosen according to the mobile operators in specific country. Clarify the configurations according to the WAP used by the operator. [16]

4.4 Verify your APN Settings

The next step was to verify your Access Point Name (APN) settings. Eventually there is need to configure your mobile device to connect to the Internet. This is done by setting up what is commonly called the APN (Access Point Name) in the configuration interface of the mobile devices need to be tested. Devices with no, or improper, APN settings will fail or generate an error when a network connection is attempted. [16]

4.5 Download the Taste MIDlet (.jad and .jar files) to the Device

Downloading a taste midlet into the phone was done in three different ways which are Bluetooth, cable and infrared. This depended on the phone capabilities or features that supported the download.

Whether using your PC's Bluetooth functionality, a cable, or the infrared port, connect your PC to the target device and transfer the MIDlet (.jad and .jar) files to the device. Another possibility is to download directly to the phone using HTTP.

Download the MIDlet can also be done via OTA. The MIDlet is supposed was placed on the server and was downloaded from there. This method can also verify the device's OTA capability under the target operator's network. In general it was assured that the .jar file will fitted the three phones. Some devices like Nokia Series 40 (MIDP1) limit the .jar size to 64k; others will limit it to fewer than 100k. Also, most devices have a low limitation on the size of all games MIDlets installed. Checking the available memory and in some cases and removing any old unused games is essential. Depending on the network and device that you are developing on, there is a small risk that OTA downloads could restrict access to Java ME APIs. Check with the respective manufacturer for the proper instructions when downloading and installing the application on the target device. [16]

4.6 Running the Taste MIDlet (.jad and .jar files) to the Device

The taste MIDlet analysed the ideal Java environment on the selected phone and then the results was reported to the worksheet below. Remember as discussed previously java is able to run on other operating system also.

The figure below shows three phones that were tested to figure out their characteristics and features involved. The summary of their featured is shown in the table below.

Figure 8: Shows three phones that were tested on the application. They are Nokia 6288, Nokia 7210 and Nokia N-gaged respectively.

The choice of the handsets above depends on their different capabilities APIs ranging from Nokia 7210 which is an older version to Nokia 6288 which is a newer version. The difference between these phone depends on there API's abilities to cater for a certain function.

The table below shows results obtained after running a taste MIDlet on these mobile devices. The significance of this table is to show the features that are found in each handset and comply with the game.

Mobile device	**6288**	**7210**	**N-gaged**
Operating system	Symbian based		SymbianOS
Platform	Nokia6288/05.92	Nokia7210	NokiaN-Gage
Screen Resolution (Canvas)	240x250	128x96	176x144
Profiles	MIDP 2.0	MIDP 1.0	MIDP 1.0
Graphics	2D,3D		
API	Mobile Media API Version (MMA JSR-135) 1.1		
Size of heap (RAM)	2048KB	210KB	10088KB
Configurations	CLDC 1.1	CLDC 1.0	CLDC-1.0
Java virtual processor speed	15.9MHz	17.6MHz	7.1MHz

Codes allotted to each key	**1** **2** **3** Not used UP Not used **4** **5** **6** LEFT FIRE RIGHT **7** **8** **9** GAME_A DOWN GAME_B **0** ***** **#** Not used GAME_C GAME_D	**1** **2** **3** Not used UP Not used **4** **5** **6** LEFT FIRE RIGHT **8** **9** **0** DOWN GAME_B Not used ***** **#** GAME_C GAME_D	**1** **2** **3** Not used UP Not used **4** **5** **6** LEFT FIRE RIGHT **7** **8** **9** GAME_A DOWN GAME_B **0** ***** **#** Not used GAME_C GAME_D

The code allocated to each key gives the developer the ultimate control over mobile games. The developer is able to map keypads, and other events to the controls on the game pad or joystick. He is also able to set up the controls the way he want, tie complex moves to a single button, or map them to any combination of controls.

Codes allocated to each key also depend on the functions allocated to them when coding. Remember when assigning the codes to the keys (key mapping), the unsigned keys, which are not used can also be mapped for other game purposes.

After receiving these results from a taste MIDlet, developer already knows what features the game will need. This does not mean that all the features in the mobile phone must be implemented in the game. The developer may also drop some contents of the game to enable smooth running of the game.

4.7 Results obtained From the Taste MIDlet on Nokia Emulators

After running a taste midlet on these phones the results are shown. This signifies the handset capabilities and features available in their profiles.

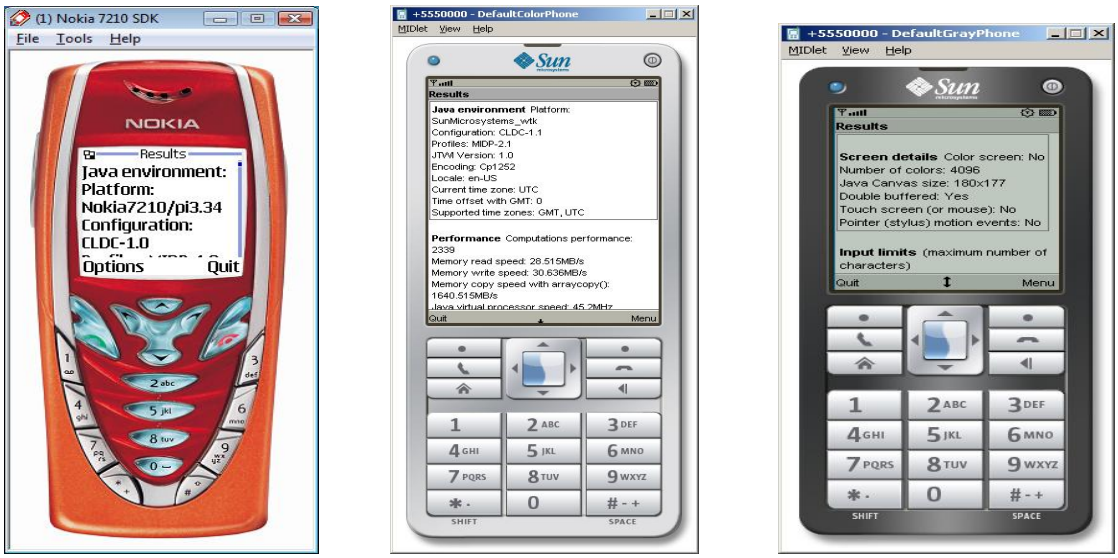

Figure 9: Shown by Nokia wireless emulators on taste MIDlet

4.8 Downloading the Game Into the Phone

Turbo Worm is a vastly improved incarnation of the classic snake games. Since this has two versions, one for MIDP 1.0 and another for 2.0, we are going to check their behaviors in their equivalent handset. [13]

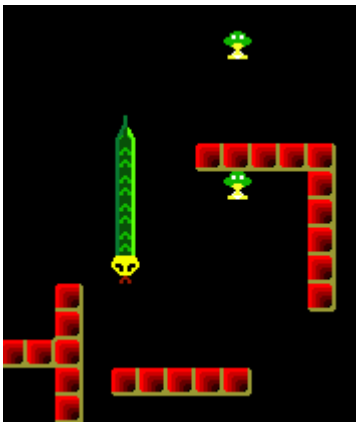

Figure 10: Shows image of a Turbo Worm game
http://www.midlet-review.com/index?content=freegames/freegames

When running version 2.0 on MIDP 1.0 handset, the game application does not launch. This is due to certain APIs that are supported by MIDP 1.0. The application error will always appear in the screen if the application does not match the APIs as shown in the figure below. The game was played a bit to on the phones to see if it is stable on the phone.

Figure 11: Shows running application on MIDP 1.0 and 2.0 emulators.

As illustrated above most of the mobile devices that are compliance with MIDP 1.0 can also support MIDP 2.0 since the MIDP 2.0 is an enhancement of MIDP 1.0.

After this stage the game is considered compatible with the phone depending on its stability. Now the game should undergo the steps involved during game testing. The next chapter discusses the steps involved in game testing.

5 Device Versus Game Testing

5.1 Game Launch

All the passed devices are ready to be to be fully tested according to their respective application. Game applications are downloaded over the air or (OTA). The handset has to be registered to the Internet. The URL address is located on the application JAD file on the OTA server into the handsets. Once the game is installed, the application icon should ideally be displayed, but any means to start the application is acceptable. Two screens, the title and the splash screen are displayed after a few seconds.

Due to the differences in mobile types and the way they were made, connection to the Game jad and jar can be done via the OTA, typing the full path that connects to the jad file or using an http connection that joins the jad via the internet. Between the stages of launching there is a loading bar which directs the user to the next stage of the game. Most handsets should ideally take less than 15 seconds in all the loading bars although some handsets take longer which creates a major problem. [16]

As shown in the figure below, the stages from downloading the game into the handset, the splash screen and title screen are shown first before the game is directed it to the menu.

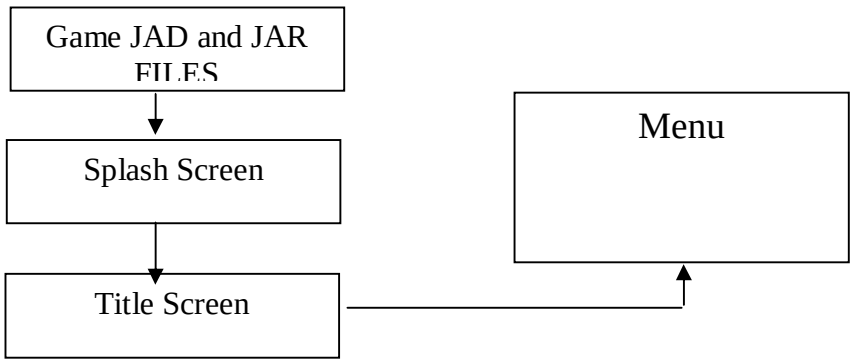

Figure 12: The figure above shows the Stages of the game launch.

5.2 User Interface

UI consistency, animations, graphics and other game related functions must be visible and workable. The user must be able to read, choose and see clearly all the application user interface. The game applications must utilize the full screen size available from the target device. Before the application is made, the game developer must refer to the screen size of the target device. This can be also be referred to the manufactures table of supported devices. The aim of the UI is to enable the user to interact with all that is included in the game under no defects. Some of the important attributes to be checked by the game tester in the user interface are explained below. [16]

Graphics play a huge part in the game. They should be visible and able to be seen clear by the user. The resolutions of the images displayed should also be of high quality. The game application should also utilize the full screen size of the total device.

The user Interface consistency should run unlimited. The performance of the game should run effectively from game launch to the end of the game playability and also to a pace programmed by the game developer.

When a gamer goes through the application the interactivity should be displayed in accordance with the user inputs. Menus and buttons should function effectively and as programmed by the game developer. The specifics of the internal design and external designs of the game must be tested. This may require a description of the possible implementations if it makes the testing requirements easier to understand such as certain theme of the game, the characters, the animation, cinematic or camera view, and so on. For example, to test the multi-directional fight action for the application, the tester must make reference to the use of the game application and also to describe how the opponents engage into their actions. [16]

5.3 Functionality

The application functionality is based on its consistency to run indefinitely. Hidden features, external incoming calls and other communications like sms and mms are involved, and the idea is to check if the application can stand all these external interferences which are not observed in the application. The application has to perform as intended even with these incoming features during game play. Hidden features like incoming calls may hinder the rate of the game play and may cause the game to crush or destroy the application. [16]

The call should be made in every location of the application including menus, settings, and inside game play. To check if the application is effective, the developer may call the phone during game play and make sure that no harm is caused to the application. When the incoming communication enters the device, the application goes into pause state. After the user exits the communication, the application presents the user with a continue option or is continued automatically from the point it was suspended at. The developer is encouraged to use the available APIs to pause and continue methods. This process is also observed when receiving an sms or mms data. [16]

Other functionalities include billing methods which are done by the respective carriers. Modern games are developed to encourage interactivity between users like.

5.4 Connectivity

Modern games that use network connections to communicate with servers, others users, and are password oriented are normally referred to as on-line builds. On-line builds are programmed to handle network connections and return a note to the user if the connection was not completed. The user has to be informed whenever the connection is not allowed, whenever there is a delay or a loss of connection and also every time the connection is closed after disconnection. Connection is also done when the user tries to communicate with the server when sending and receiving high scores. [16]

When the application uses the messaging capability of the device such as sms and mms, it must be able to send messages correctly and notify the user in case of any error message. Other connections include Bluetooth and infrared. In off-line builds the connections however, are non networked applications that simply use the device platform to run the game software. The games may be installed over the air, or they may be induced onto the handset with a cable.

All modern mobile games have network capabilities and must be able to handle network connections. All networks in mobile applications are able react in three ways. First, when the network is allowed and secondly, to set the network connectivity from the device settings to "Not allowed" or disable the Internet profile, or try to connect the network via the application and see how it reacts. Thirdly, when there is a network delay and the loss of connection brought by latency or inefficient supply of network.[16]

When the Network is Allowed

When the network connection is allowed in the mobile applications, it means that all connections are able to work with no fault. The user is able connect to all network capabilities.

When Network Connectivity from the Device Settings is "Not allowed"

The tester is advised to halt network connections from the mobile device and go to the application and enable connections after stating the application. A note should be sent to the display that error in connection.

Delays and the Loss of Connection

The application is able to handle all the network delays and loss of connection. The user should start the application and enable the network connection from the application, put the phone in a place where there is no network and observe the results. The application

will work until time out and then give an error message to the user indicating there was an error with the. This could be also caused by latency or much network connection synchronizing at the same time.

5.5 Security

The security of mobile games includes protections of passwords, security prompts, encryption, and security warnings incase of a mislead. In places where the passwords or sensitive data are inputted, input your data and see how if behaves on the screen. A tester can also try and put some sensitive data and see how it reacts. Also there is no possibility to send the game to another device via any connection when you are downloading the game via the carrier portals. [16]

5.7 Sound

Sounds should play at the beginning of every application without being distorted. Options to turn on/off the sounds should work. One sound setting is in the mobile device and another in the game. The default sound should be in the mobile device which means that when the mobile device sound if off, the whole game sound is also off. When the developer turns on the phone sound and turns off the game sound, then the game sound should be off.

5.6 Localization

The localization team is responsible for handling the misuse, misspelling and character sets of the application. The carrier language specifications are normally considered as the first priority. Spelling errors, technical text errors, translation accuracy and language flow have to be taken into account too. There are two kinds of localization testing which are based on the application languages. [16]

Single Language Applications

Since the application is based on one default language, the localization team is able to work with the developers to ensure that from the application's launch to the end, no language misuse will take place. Normally low-end handsets are used in single application because their memory limitation is limited in a way that they can handle single language.

For any application JAD/JAR pair using only one language the entire criteria will be performed. For other areas of the application, localization team is responsible to ensure that all localization criteria are respected throughout the application. An approval granted to any application that does not meet the localization criteria is removed. [16]

Multilanguage applications

In case the application JAD/JAR pair incorporates several languages, the application will be tested using English by default (or any other language if English is not present). Multilanguage applications require more time for. [16]

5.7 Game stability

Game stability can also be stated as the process of validating and verifying that a game application meets the business and technical requirements that guided its design and development, works as expected and can be implemented with the same characteristics. After running all this tests, the game should not crush or freeze at any all any of this stages. The application should also respond to the user input without any problems.

6 Problems Involved in Compatibility Issues

Mobile Devices

Nowadays there is a big difference in the mobile game business from the wide range of devices and platforms that the games are commonly deployed on and their idiosyncrasies. To further understand these issues it is important to understand how the mobile device has developed. The current generations of mobile phones have the power of making this viable. [16]

Write Once Test Everywhere

J2me applications are said to be written once and run elsewhere, but that's not only the case. These applications are tested to different variety of mobile devices which makes the testing process cost effective and time consuming.

High Speed, Low Latency Network Connectivity

The increase of high and sophisticated games have enhanced the rise of high speed CPU's which makes mobile games become even more complicated for developers and users. Modern mobile games like any other PDA have increased network connectivity induced in their applications. This increases low latency.

Lack of Standardization

With the diverse increase of mobile technology, mobile companies and game developer are finding it hard to standardize steps which should be followed by all. This creates a diverse and complicated way of dealing with technology issues.

Device Diversity and Fragmentation

Developers and testers now have to deal with tons of different devices in and coming to the market. This creates confusion and time wasting to go through their features and characteristics. Also creating multiple applications for multiple devices increases the development period.

Increase of Technical Knowhow

New programs, versions and software are coming to the market faster than expected with a different set of platform. Before one of these has been fully commenced, there is need to go and learn the new or added features of the software.

Wireless Carriers

Wireless carriers try to market different services by customizing supported devices and their software. It requires a lot of work in customization of J2ME applications for each carrier and device.

7 Conclusion

The study in this report discusses issues to be considered before porting game applications to mobile devices and steps to be followed after inducing the game to the device. With the information provided in this report, the game developer will hopefully be able to distinguish these issues and make affordable compatibility choices. The increase in technology innovation for mobile games will always allow developers to rely on manufacturer's mobile device features.

J2ME is also supported on mobile devices worldwide by almost all manufacturers, and across all air networking standards. This makes it one of the most common platforms that game developers use. However, the game development process using J2ME can be quite complex because the technology was designed for constrained devices. Developers have been relying on J2ME platform for many years and manufactures are always looking for new platforms and operating systems to run their devices. The only aspect left is to find a way developers can incorporate their products with these devices.

As a result, although J2ME exists as a single product name and shares a fundamental core, there are subtle differences in its implementation on various handsets, at least between handset families. This is one reason why a particular game can sometimes run on one handset but not another, even if the handsets are made by the same manufacturer. Also the characteristics based on this study show that different configurations depict different ways of solving the challenges.

The results of this study showed different results from different mobile handset depending on its application interfaces. The results also showed the compatibility compliance on the game application with different mobile handset. These compatibility heuristics are often used to determine whether a game must be modified to function with a popular component and indicate the game's minimum system requirements for marketing purposes.

References

1 2005 Mobile Games White Paper [online]
URL http://www.igda.org/online/IGDA_Mobile_Whitepaper_2005, Accessed 20 January 2008.

2 Mobile Audio Technology, Report and Recommendations [online].Los Angeles CA. Midi manufactures association; 2005-2007.
URL http://www.iasig.org/pubs/mawg-rpt.pdf, Accessed 20 January 2008.

3 Lam Jason. Considerations for mobile game development [online]. 07.January.2003.
URL http://archive.wirelesspronews.com/ 2003/0701.html. Accessed 20 January 2008.

4 Söderlund Tom.A different game, proximity gaming on Game Design [online]
URL http://www.differentgame.org/detail.asp?item=672. Accessed 15 February 2008.

5 Three simple steps to localize[online].
URL http://www.sisulizer.com/support/localization-traps-character-sets.shtml Accessed 20 June 2009.

6 Crooks II Clayton E, editor. Mobile device game development: Game Development Series; Charles River Media, Inc; 2004.Rockland, MA, USA.

7 Porting J2ME games to flash [online]
URL http://www.adobe.com/devnet/devices/articles/
porting_ j2me_flashlite_ 03 .html. Accessed 12 December 2007.

8 Juntao Michael and Sharp Kevin. Developing Scalable Series 40 Applications: A Guide for java developers. Nokia Mobile Developer Series; 2004.

9 Link Johannes with contributions by Fröhlich Peter. Unit Testing In Java. How tests drive the code. Morgan Kaufmann; 2003.

10 Hayman Craig (IBM). Device fragmentation of mobile applications. [online].
URL http://www.comp.nus.edu.sg/~damithch/df/device-fragmentation.htm, Accessed 20 February 2008.

11 Korhonen Hannu, Koivisto Elina M.I. Playability heuristics for mobile games [online]
URL http://research.nokia.com/files/p9-Korhonen-Authors Version, Accessed 20 November 2008.

12 Taste Phone Server version Alpha 2 [online]
URL http://www.club-java.com/ TastePhone/ TastePhone.jad , Accessed 12 January 2008.

13 Mobile gamer. free mobile games for J2ME supporting handsets [online]
URL http://www.midlet-review.com/index?content=freegames/freegames, Accessed 15 March 2008.

14 Java.net The source for java technology collaboration.
URL https://tastephone.dev.java.net/. Accessed 11 January 2008

15 SNAP_Mobile_API_Compatibility_Test_Instructions [online].
URL http://www.forum.nokia.com/info/sw.nokia.com/id/f40dd61b-e0a6-4480-85c7b58a6242dcf9/SNAP_Mobile_API_Compatibility_Test_Instructions_v1_0_en.pdf.html. Accessed 27 January 2008.

16 Unified testing criteria for java(TM) technology-based applications for mobile devices [online].
URL.http://javaverified.com/docs/UTC_2_2.pdf, Accessed 27 January 2008.

17 MIDP: Mobile Media API Developer's Guide Version 2.0; October 31st, 2006 [online].
URLhttp://www.forum.nokia.com/main/resources/technologies/java/documentation/Java_ME_developers_library.htm. Accessed 23 January 2008.

18 The future of game development: New skills and new and new attitudes.
Part 1: Mobile games By Marc Mencher [online]
URL http://www.gignews.com/careerfeatures/skills04part1.htm, Accessed 15 February 2008.