C O V E N T R Y
U N I V E R S I T Y

School of Engineering

Full Course Title: European Engineering

Final Year Project

Project Title: Java Programming for Mobile Phones

Author: Marcin Radlak
Supervisor: Dr Qin Zhou

Submitted in partial fulfilment of the requirements for the Degree of Bachelor of Engineering

Date of Submission: 20 April 2006

DECLARATION

The work described in this report is the result of my own investigations. All sections of the text and results that have been obtained from other work are fully referenced. I understand that cheating and plagiarism constitute a breach of University Regulations and will be dealt with accordingly.

Signed: **Date:**

ABSTRACT

The aim of this project was to develop a java application or a game for mobile phone. It covered setting up the correct development platform, describing the problem in human language and implementing it in programming language - finally testing the solution on the mobile phone which had to be communicated with the computer at first.

As my knowledge of J2ME platform was very poor at the beginning, I have set up a goal which was to gain programming proficiency in it. Secondly, other goal was to develop standalone application that can be run on the mobile device and not only mine, but also on my friends' mobile phones. It had to be fun, enjoyable, making free time pleasant, also serving player with dose of intellectual experience.

Throughout the process of development I intended to widen my skills not only in J2ME but also in Project and Time Management, research and report preparation (graphical and methodological resources processing).

Every goal above was a component of the final personal goal, which was to prepare an outstanding final report that will fulfill all the requirements, present how much work I have put into development and will be the first professional publication of my own reminding me the moment of graduation and summarizing my university work.

Throughout the following chapters I will show in details how did I come up with solution. Every chapter describes a small bit of the big task. And this is how I have completed the project, by completing small steps which joined together at the end produced this report and enabled me to fulfill all the objectives.

TABLE OF CONTENT

INDEX OF FIGURES

ACKNOWLEDGMENTS

I would like to thank every person who helped me with advice that lead me to completion of this project: specially my supervisor Dr. Qin Zhou for leading me through my work, every lecturer that via many courseworks taught me how to write correct reports, my friends for testing the application I have developed, authors of the websites about mobile devices programming for tips helping me resolving technical problems and Martin. J. Wells for the book “J2ME – Game Programming” which was a source of some interesting ideas.

INTRODUCTION

Before the project will be presented, moreover – before it was even started, it was crucial to think about the key aims and objectives, this big part of work must fulfil. In general, as it is presented in document called "Information for Project Students" there are four key aspects explaining outcomes of the task:

- to become an "expert" in a specialised field of study
- to develop skills and experience in report writing and presentation
- to integrate knowledge gained in different course modules
- to exercise initiative, creativity and imagination.

All presented above reasons for the project were the most important to fulfil academic requirements. As these are very general and can outline the work needs to be done, following them, specific goals for this particular project also had to be created.

At first when the subject was known – Java programming for mobile phones – the biggest challenge was to decide what kind of programming it has to cover, for the project to be consistent, leading to established goals. Application programming is very interesting part of software development, but as the mobile devices are very limited, game design was fascinating choice giving amazing opportunities to use creativity, initiative and imagination. Everything because of the power of mobile phones – they are to weak to run impressive application but strong enough and popular to develop fascinating, addictive and enjoyable game. Additionally, many of handsets holders use them to spend time in pleasure playing their most liked games. Everything about the limitations of small devices programming is covered in Chapter 1: Mobile devices and Java. It describes brief history of cell phones and also foundations of Java are presented.

Developing a game was a step after essential preparation about this process. What types of games can be developed, their limitations and general characteristics followed by tips which make a game being good together with available developing platform is described in Chapter 2: Games programming and available platform.

These two chapters were to present a basic knowledge of the topic and to build foundations for the rest of the work. It also caused to invent goals, that had to be achieved apart from the general ones. These were:

- learn how to program using J2ME
- get familiar with programming environment for mobile devices
- create standalone, fully working game
- produce outstanding report

Presented purpose of the project joined with reasons described at the beginning of Introduction create a very strong driving force to complete the project. Because having those two specified (reasons and purpose), actions needed could be developed much easier. It resulted in the plan for the whole 2 terms presented on Gant's Chart attached to the report in Appendix 2.

Results of the actions are described throughout the report, being organized by the subject they regard. In Chapter 3: Chosen application description the theory of developed game is presented. All the rules are explained to create general overview of the game for the person who has never played it before. It is described using simple language, by which rules should be understood.

Next Chapter 4: Inside the game - coding was written to explain biggest issues of the code and is kept in form of explanation rather than deep analysis. If deeper analysis is required, appendix 1 contains the source code for application.

Following Chapter 5: Inside the game – playing and testing, presents the game analysis from the user point of view. Every option of the front end is explained and supported by screenshots of the game screen also explained in detail.

Final Chapter 6: Discussion and further works is the finalization of the whole report. In this place most of the achievements are described, failures presented and analyzed. Finally, this is a palace for future works recommendations.

Chapter 1

Mobile devices and Java

History in brief

Mobile devices market has noted extremely high expansion during last two decades. Since developing first mobile phone of a size a big suitcase, expansion in electronics and telecommunication lead to minimization of handsets. It can also be easily noticed that almost every person around is familiar with mobile phones. This was caused by amazing popularity of GSM networks which gave the possibilities never known before. Radio communication required a lot of hardware: radio transmitter, huge sized aerial and what kept it all importable, source of electric current. It was one of cheaper way to communicate with friends, family and even with all the world. But because of its limitations, could not satisfy demands of the population.

Because of enormous desire to cheap, common and reliable communication, companies producing GSM phones had a challenge to match: produce mobile phones which everyone could rely on, use and buy for affordable price. This was achieved very fast, when phones started to fit in hand. They still were very expensive, but business men required connection with their offices, to keep maintain companies. But only mass sale of mobile phones could bring significant profit to GSM companies. On the other hand, the market had to be extended while competition started to grow, offering new technologies, better and more reliable devices. The most important benefit, what GSM technology was giving to people was

freedom – tightened at the beginning because of low network coverage – but as business started to be profitable, the network of transmitting stations was gradually being expanded.

As the prices were constantly dropping down, the number of users which could afford buying handset was growing very rapidly. More companies started to realize, that it is becoming large operative field, which in future can generate tremendous amounts of income.

Telecommunication companies started to compete with each other, to win more customers. Cheaper calls were not a good method, because caused them to earn less, while loans taken to build relay stations had to be paid off. The solution they developed was to offer headsets for lower cost. It created a pressure for manufacturers to develop attractive to the public phones. And this was the time when mobile phones' life as only a communication unit was over and the era of multi functionality has begun. Features that started to be offered grew until these days dramatically resulting in devices which are personal organizers, dictaphones, timers, cameras, camcorders, diaries, data storage, etc. with option to make a phone call.

Shown phenomenon cannot be surprising to anyone. As the technology goes further, people expect to be able to use it not only listen about it. They all want to be part of innovations offered, no matter what the age is or race. And what is more important, mobile devices has become part of everyday life and many people could not imagine life without this small gadget. To make the matter clear – many are addicted to them. And if addiction exists, some people will use it for achieving their own goals. Many users want to be proud of their mobiles, so many facilities to meet this need started to grow very fast: ring tones, wallpapers, screen savers, applications and games.

Mobile phones – performance and capabilities

As the first three add-ons (ring tones, wallpapers, screen savers) need small effort to prepare and use them, what can be done by downloading simply converted files from PC, last two – applications and games – have to be specially developed. It is not possible to just use converter which will produce a file ready to upload and launch on a mobile. The complexity of a problem lies in the limitations of a hardware. These limitations are being lowered as the years are going by, but long time is needed until the device of the performance equal to standard, these days PC, will be designed and produced. Everything is about size of the device, which cannot be reduced more yet. Even having less powerful computation power to launch top desktop PC game on a mobile it is still good enough to launch simple applications and simple but absorbing games, which gradually are becoming more complex.

The biggest problem existed and was solved just few years ago was one of the biggest limitation: if anyone wanted to write a game for a mobile phone, had to know its assembly language and if he wanted to upload to its device, had to use specialist software, which only could be found in service. Other possibility was to try to persuade manufacturer to include the game in the newly produced mobile phone, what was far more difficult (almost impossible). It caused this business to stay in stagnation, because it was not worth effort that was required. To say nothing about the efficiency of the development process and huge costs it required to be covered. And with constant improvement of mobiles' CPU power, customer started to demand more from their pocket gadget than just device for making calls.

Hardware independence

As a solution a hardware independent platform had to be developed to eliminate those limiting factors. And looking at the internet at that moment, which was going to be hardware independent (once the content is done – it can be seen from any computer fitted with browser), group of experts brought up to existence by Sun in 1990 has announced their results after long time of hard work on this problem. Taking best solutions from every programming language (C++, SIMULA) programming language called OAK was developed.

It was not just one of the next programming languages. Person responsible for the project, James Gosling, together with his team created a full programming platform, that was fully hardware independent. There was just problem with popularity of the programming platform, but sun started to improve the technology and finally, the mile stone of Java (one of the version of OAK) has been done – Netscape decided to integrate Java with its browser what created a place for this software. Since then, new versions containing fascinating features were published very often, keeping the language simple, fast, bug free and most important – staying hardware independent.

Micro version of Java – J2ME

Sun was finally in a possession of very powerful tool that was very popular. Despite of it, they decided to expand into the market that was very new at the late 90's – mobile devices. As it was the area which extremely required what Java was. Writing about limitations for applications or games developed for mobile phones, Sun could offer a relief for mentioned restrictions. But could not be applied in the version that was designed for standalone PC. That was a moment when Java Micro Edition was prepared.

J2ME is a programming platform which revolutionised mobile, lower performance devices. It is based on Java Standard Edition, but had to be fitted into a machine with resources much smaller than J2SE required. Sun decided to choose the most important components and cut the size of the platform reorganizing its structure what resulted in increased performance which required smaller resources and could be applied to the mobile devices.

J2ME activated mobile programming market. It was now possible to develop a software and not to worry how to launch it on number of different handsets. It made possible to produce hardware secure software, easily installable and with many other features which made the platform famous and implemented in almost every single mobile phone.

How all Java versions suits devices is illustrated below.

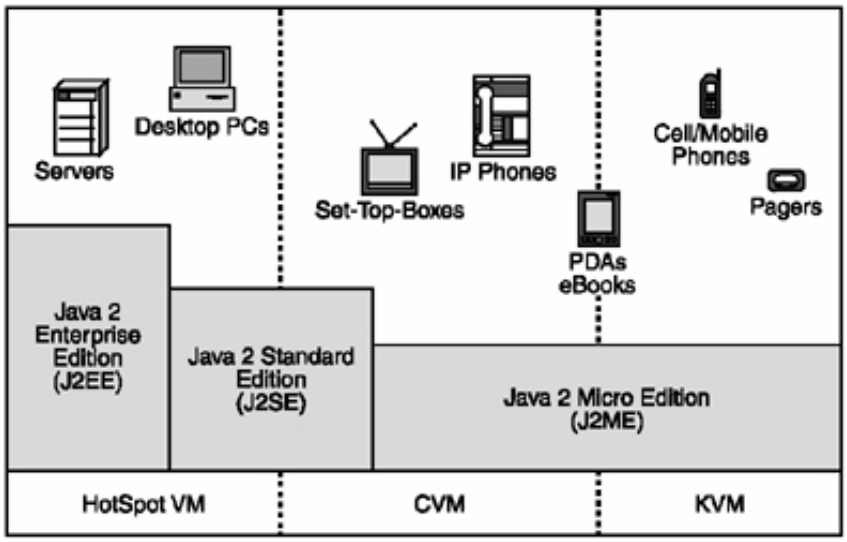


Figure 1.1 How different editions of Java suit different devices [1]

Java architecture [1]

Java ME is based on configuration profiles (CLDC, CDC). They define minimum requirements which are necessary to run a game. These relate to memory, processor, connectivity and some other like low battery consumption. It does not specify screen requirements or input which is done by specific device class profile MIDP (Mobile Information Device Profile).

Very important to mention is also security model which is covered by CLDC and may be divided into Virtual Machine Security and Application Security. The goal of the virtual machine security layer is to protect the underlying device from any damage executable code might cause. The reduction was achieved by the removal of the iterative dataflow algorithm

from the in-memory verification process. The price is that additional step known as pre-verification has to be done to prepare code for execution on the micro device. The result of this process is the insertion of additional attributes into the class file. It is shown on Figure 1.1

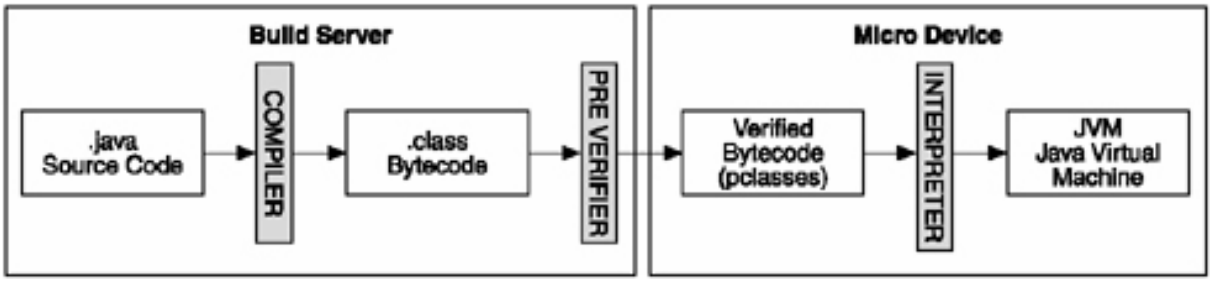


Figure 1.2 Process of program build and execution

Application security was achieved by using concept of sandbox. As it can be seen in Figure 1.3, program code has restricted what's available in the sandbox environment. The CLDC defines a list of exactly what can be execute. Protection is also in place so it is impossible to change the base classes that make up the installed API on the device – core classes. The CLDC specifications mandate protection for these then.

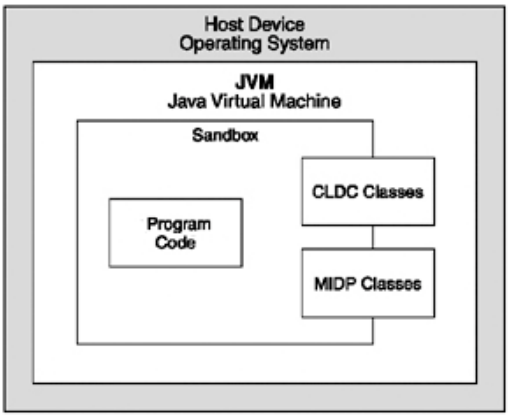


Figure 1.3 Application Security Model

My choice of programming platform

With all the features described above J2ME is very powerful, yet simple programming language. It is on the other side very widely implemented into almost every mobile phone. No wonder that this programming platform has been chosen for this project. It allowed to develop a program, which was decided to be a game and later on to copy and test it on my and friends' mobile phones giving very interesting opportunities.

Chapter 2

Games programming and available platform

Limitations

Usually developing a game for PC users (unless it is very complicated 3D challenge) there is no need to worry about memory, CPU and screen limitations, which are the biggest issues for Mobile Device programmers. The code that is developed should be optimized deeply with use of every known technique to improve efficiency. Following section is to describe what are the biggest challenges developing mobile phone game

Screen

Mobile devices as its name says are mobile. They are mostly as small as possible to easily fit into pocket. And a screen size cannot be larger than the device. It is usually in the range of 100200 pixels. The brightness is limited together with number of colours. Also the sizes for different devices varies very widely so this has to be kept in mind. Pixel response for old type screens may be also very slow, causing blurring effect for action or fast changing games.

Storage

There are two types of memory limitations. First is the limited size of RAM, which is necessary for game during a play. It is usually between 100 – 500 KB and is growing very rapidly. Second is the space for games, store memory. It is usually limited to approximately 50 games that can be stored in the phone at once.

Controls

Most of mobile phones have similar keyboard, but there are plenty of cases where keys layout is not similar to the one that development process was tested on. Thus that, it is hard task to develop very universal game. Additionally, some of the MIDPs do not allow simultaneous key presses handling so developing action games is very limited.

CPU

Much lower performance simply because such device does not require very high computing power. Because of it is difficult to develop mathematical programs or efficient 3D game but hopefully this will be improved in close future.

While developing a game all the previous issues should be kept in mind to avoid limited scalability. With good knowledge of what can be done and what not, omitting major factors causing problems is much more easier during the development rather than while testing and debugging.

Types of games

There are many types of games available to be developed for mobile devices, but it should be kept in mind, that some of them are easier – others much more difficult to adopt to the small mobile device. Because of this reason next section explains some of the major game types as follows.

Action Games

These are mostly games that require fast response from the player, but more important, need much more resources than other games. Game screen is usually constructed of sprites and background images, which (if good quality) are bigger sized so require more memory. Also, to retain ability to run game on different devices, it is required to fit in to minimum requirement of CLDC. Depending which CLDC will be used – MIDP also has to be considered.

Puzzle/Mind Games

These types of games covers jigsaws, word games, memory games and sliders. They do not contain to much action but still can be very popular and addictive. These games also require some brain effort and in my opinion are very enjoyable to play because of making challenges with computer or with just simple logical problem which does not require AI implementation to create machine player.

Adventure Games

Adventure Playing Games (RPG) rely on exploration of new environments, being a key motivator for player. This is limited for mobile phones as the new environment memory for new graphics. Fortunately, this problem is solvable, but need i.e. effort to prepare low sized graphics and other resources.

Strategy Games

Strategy games are either real time (RTS) or turn based. The settings and perspective can vary, but generally strategy games involve the acquisition and tactical deployment of available resources for the greatest gain by doing this against other players (even AI ones). Simulation games also fall into this category. These types of game involve the management of deeply complex system and is about understanding and affecting the nature of the simulated system than it is about employing a proactive strategy

Traditional Games

These games, such as gambling, card, board and dice games and because of its simplicity are very popular throughout the users. They are mostly static, without much action, but keeping entertainment in mind can make them very enjoyable.

What makes a game being good

This is the core question that every game developer should ask himself. The question is made of five requirements: fun, addictive, compelling, outstanding and above all, commercially successful. And following in the following section it is shown what are the key solutions to meet mentioned requirements.

Goals

This is the part of the game that makes people want to play it. They must be challenging, but achievable. It is a good practice to reward player after some major progress he has done.

Balance and completeness

The player has approximate time that takes to achieve next level in the game. If it is to hard or to easy, game is getting boring or frustrating. To be successful developing this, giving a game for beta testers is easy way to find out what others think about it.

Originality

This should result in producing amazing experience for the player. But most of them like to play game that is similar to some others, that's why most of the games are similar to one template. The originality is achieved together with details rather than general concept. To be successful with a new template is very challenging task

Technology

This must be measured to customer expectations. Even best 3D game will be dismissed by players if it has low entertainment value. This has to be implemented in the game that way, that players will not notice it so it needs to run smoothly, look good, and deliver the game's entertainment value without a problem for player..

Available development platforms

With all the introductory knowledge described preciously, it was the time to start developing my own game. J2ME program can be developed using many IDEs:

- Borland JBuilder X Mobile Edition
- IBM WebSphere Studio Device Developer
- Research In Motion BlackBerry Java Development Environment
- Sun Java Studio Mobility
- NetBeans IDE 5.0
- Eclipse J2ME plug-in
- Nokia Developer's Suite for J2ME

They all have advantages and disadvantages regarding features they offer and ease of integration with third part components i.e. mobile specific emulator.

This part of the project describes what was used, what are the features and requirements for the developer, and why and what did I use as my development platform.

Netbeans 5.0 description

NetBeans IDE 5.0 introduces comprehensive support for developing IDE modules and rich client applications based on the NetBeans platform, the new intuitive GUI builder Matisse, new and redesigned CVS support, Sun Application Server 8.2, Weblogic9 and JBoss 4 support, and a lot of editor enhancements.

Here are some of the interesting features in this release:

Developing NetBeans Modules, Matisse GUI Builder, Servers, Web Frameworks, Web Services, Editor Enhancements, Code Completion, Refactoring, Version Control, Debugging, Other Usability Improvements, New NetBeans Add-on Packs.

NetBeans Mobility Pack 5.0

NetBeans Mobility represents the most comprehensive, free Java™ ME authoring solution on the market. Based on the award-winning NetBeans platform, NetBeans Mobility 5.0 provides an unmatched set of tools, utilities and wizards for mobile Java developers of any level. Its main features are:

- Improved Mobile Client to Web Application Connection Generator – easily access Web services and other server-side data from MIDlets via servlets.
- Support for Java ME Web Services (JSR 172) – write applications that access Web services directly from the phone.
- New custom components in the Visual Mobile Designer – new components enable a developer to add tables, wait screens, and splash screens to the user interface.
- Improved preprocessor support for device fragmentation – develop code for any number of devices with a single code base.
- Edit generated code – guarded blocks are now more flexible, enabling the developer to customize code generated by the Visual Mobile Designer.
- Improved emulator support – easily add custom emulators to the IDE.

Sony Ericsson K750i manufacturer description

“Form follows function in this attractively designed phone with a compact body which cleverly includes dual fronts, one for the phone and one for a real camera look and feel.

K750i offers the latest in imaging, advanced messaging and connectivity technology, with a rich offering of multimedia and entertainment functions. This includes, for example, taking pictures with great picture quality with the built-in 2.0 Megapixel camera and for true camera comfort, the user interface is turned horizontal when using the camera. Easy-to-use imaging communication provides a dedicated camera button to minimize the number of steps for taking and sending a picture or video clip. There is optimized memory for video communication with up to 38 MB of built-in memory for storage of content such as pictures, music, ringtones, themes, games and video clips. The Memory Stick slot is placed just beneath the keypad and a 64MB Memory Stick PRO Duo™ is included together with an

adaptor in the kit at purchase. Memory Stick PRO Duo™ supports up to 1 GB memory capacity.

A powerful gaming solution for Advanced Java™ 3D with cutting-edge graphics, multiplayer games and a large 1.8 inch 262k TFT colour screen lets the user get the most out of the phone when technology meets design and creates a friendly user atmosphere."[3]

Sony Ericsson Development Kit SDK v2.2

While the mobile phone that was available for me is manufactured by Sony Ericsson the decision was made to use original software supplied by this producer.

The other reason why it was worth installing this software is a feature of testing written application directly on the mobile phone. It is called "On device debug tool" and require direct connection between mobile phone and the computer. This was very useful but not always necessary, so here came up another valuable feature that was offered – mobile phone emulator.

During the process of development it helped greatly, because of almost perfect phone imitation. Some problems deploying later versions of the game were encountered, but these were very little which did not affect the overall process.

To get better characteristics about mobile phones it is suggested visiting Sony Ericsson website for developers [4]

Reasons of choice

After beginning my development with Eclipse IDE and spending some time to set it up without success, I decided to move to Net Beans IDE 5.0. It was still beta version but its interface and ease of setup helped to decide about developing the game in this environment. After all, even being beta version, had many features not present in version 4.1 which was the latest stable version that time.

This IDE also did not cause any major problems while integrating it with Sony Ericsson Development Kit v 2.2. This SDK I had to use because target mobile phone was Sony Ericsson K750i fitted with Java configuration CLDC v 1.1 and MIDP 2.0 profile. That means that very powerful environment was available to develop for as mentioned configuration and profile is one of the newest standard.

Chapter 3

Chosen application description

Decision justification

With knowledge of technology and programming environment type of application it was going to be develop had to be chosen. Choices varied between application and game. As the methodology of game programming was described earlier, it was easy to assume that the game was chosen to be implemented.

This decision was made because mobile market is very new type of entertainment area. Writing a game will generate much more fun and pleasure for others. It will also make easier to find testers for the game than an application. In addition to this it is the market that demands constant improvements from hardware and creates new challenges for manufacturers. Furthermore it causes IT industry to grow in very fast way making the most of available hardware.

Finally, programming for mobile devices is a new challenge which I wanted to accept to learn the latest technology.

Type of game

But deciding to write a game was followed by another struggle to decide in detail what is it that was going to be develop. There are many action games on the market available to buy from vendors but there is a lack of strategic or mind games. I personally like much more to challenge my brain having a moment of free time (i.e. waiting in a queue) rather than

thoughtlessly play in action game. Brain challenge is great way of relaxing and mind training. In the age of the information overload, people are subconsciously served with data that do not require thinking, making our brain less efficient.

For all above reasons, the decision was made to develop mind training game which was finally called Mobile Soccer.

Mobile Soccer – the rules

Game, as its name says, is a form of football match. Its rules are therefore very based on the real football game rules. They will be described now in detail.

Following Figure 3.1 shows the start screen of a game. It is assumption that this is a football field which is constructed of small squares formed into bigger rectangle (10 squares high x 8 squares wide).

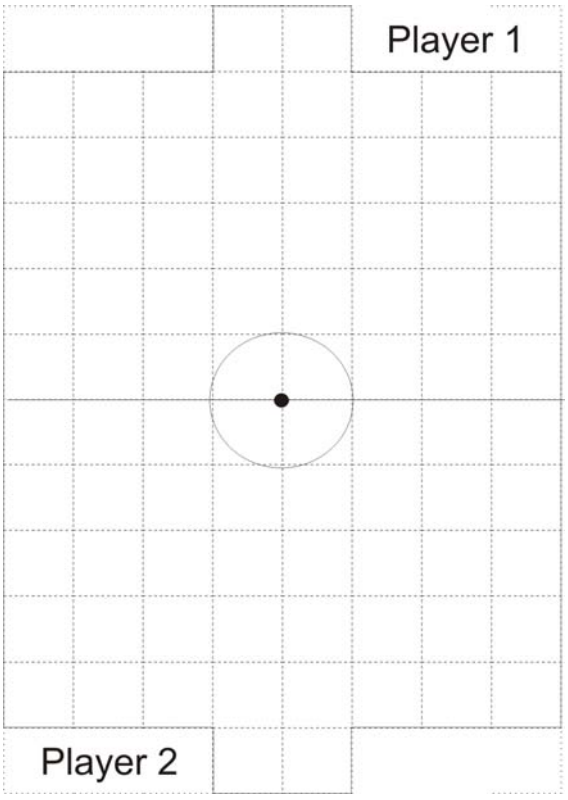


Figure 3.1 Start screen of a game

Black dot in the middle is assumed to be a ball. Presuming that Player 1 starts the game, he should direct it to the Player's 2 goal. All 8 directions are available to go to. This is shown in the following set of game screens on Figure 3.2

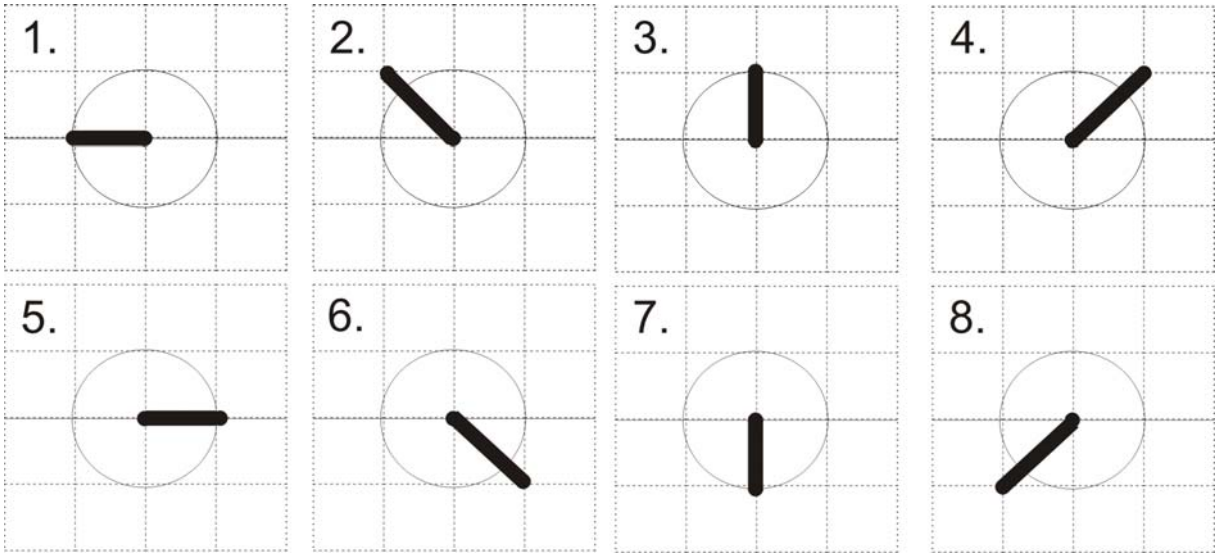


Figure 3.2 Direction available during one player move

Obviously, the player 1 (as it is a player which started the game) should go towards opponent's goal, and should get there before the opponent reaches his goal.

But trying to achieve game objective (score a goal) would be not very interesting if only mentioned possibilities were possible. So there is one more additional rule, which makes the game very interesting, suprising and pleasurable to play: if the player's end position is in the field that has a line already drawn, he gets another chance for another move. To make description clear, one possible situation taken from game was prepared and is shown in following Figure 3.3.

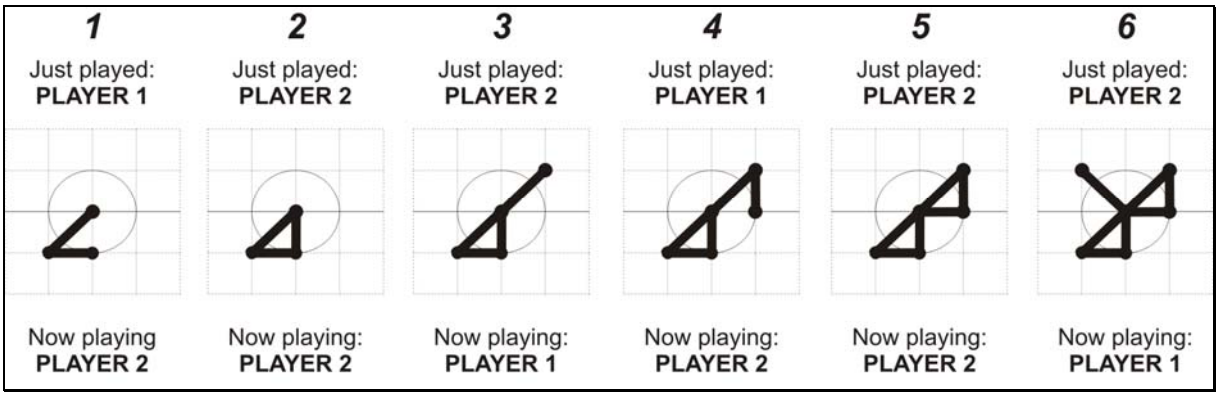


Figure 3.3 Possible game situation

Stage one on the above figure shows that the Player 1 has just finished his move, and as it ended in the empty field, its Player's 2 turn. He moves one up (direction number 2 from Figure 3.2) and as his move ends, where the line connects to other line (Figure 3.3 – Stage 2) he can make another move what is seen on Stage 3 (Figure 3.3). This is now Player's 1 turn, as the previous move done by Player 2 has ended in the empty field. Next stages of a game

shows possible situation of a game between two players that also illustrates what moves are possible (allowed), how to play and obviously shows how the game is organized.

The winner is a Player which finished his move on the end line of the goal. Example of the winning move of the Player 1 shooting a ball into Player's 2 game is shown on the following Figure 3.4.

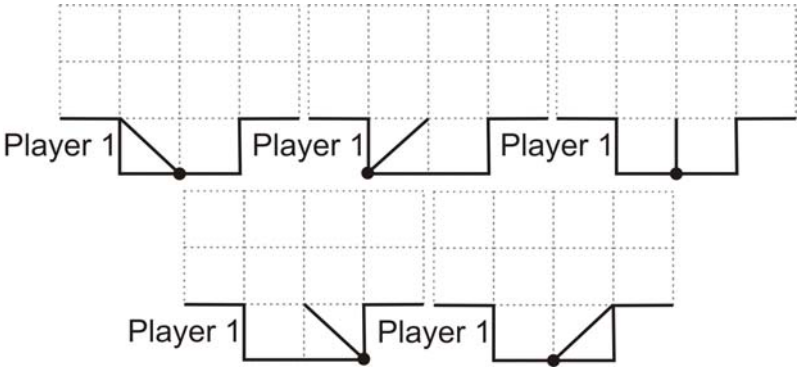


Figure 3.4 Example of finishing game position

Obviously, this situation was only for Player's 1 goal. Opposite situation will be for Player's 2, but it is not worth describing (drawing) this for the second time.

Finally, if the player gets to the band one square close, he can use it as already drawn line, so bounce from it. It is shown on the Figure 3.5.

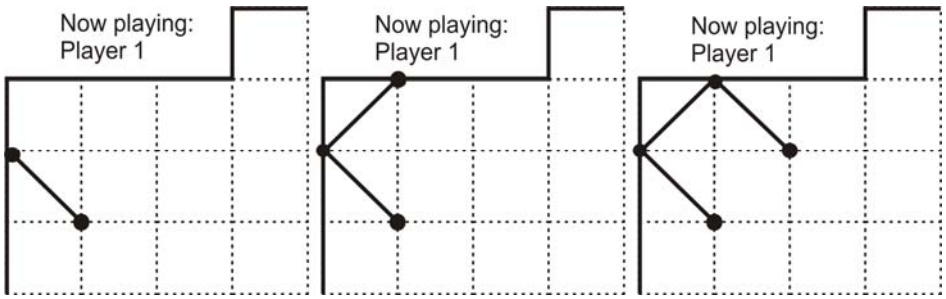


Figure 3.5 Example of bouncing from borders

This are all the rules of the game. Knowing them it is time to start playing it.

Mobile soccer – more advanced game states

Following Figure 3.6 shows the sample game state.

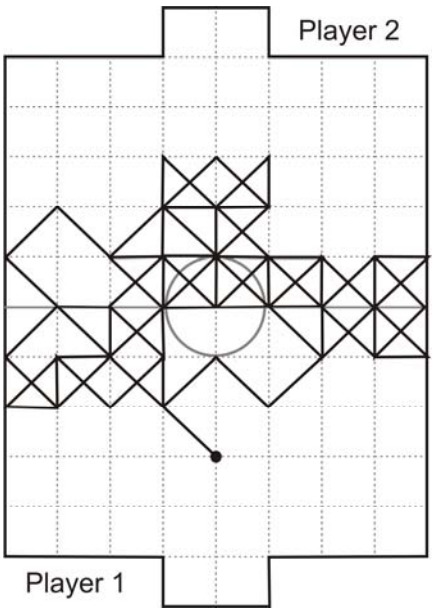


Figure 3.6 Sample cut off situation

It is player's 1 turn, so obviously he would like to move ball as far as possible from his goal and also try to get as close as he can to Player's 2 goal. But this will be not possible. Player 2 with his turn, blocked the path to his goal. So there is no chance to score a goal. But the game is still not lost for him. There is also a possibility to force opponent to make wrong turn. What is meant by that is i.e. to direct opponents moves into the situation where there is no possibility to make correct turn. This situation may happen if i.e. player gets to the corner (see Figure 3.7)

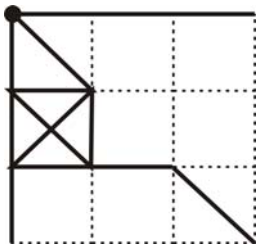


Figure 3.7 Sample ending situation

It is clearly seen that no other move is possible and using this kind of tricks, even player who was cut off from opponents side of the field (Figure 3.6) has still big chances to win.

Finally there are also some tricks to move faster to opponents goal rather than responding for his own moves. It can be done by that kind of moves, when opponent is unable to move using previously drawn line. Some examples of the situation are shown below:

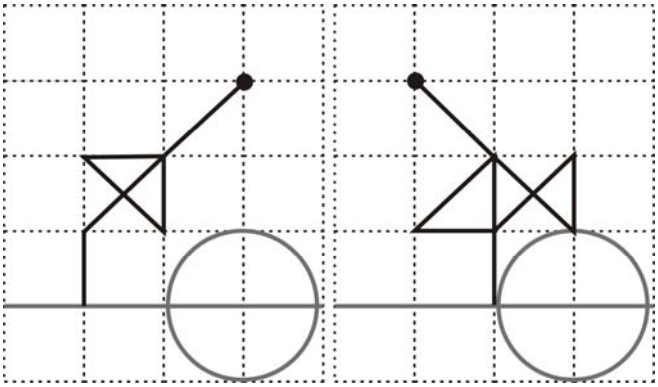


Figure 3.8 Sample moving forward situations

The last set of situations is when there is under goal situation. There is always chance to avoid loss when the game is almost lost (Figure 3.9).

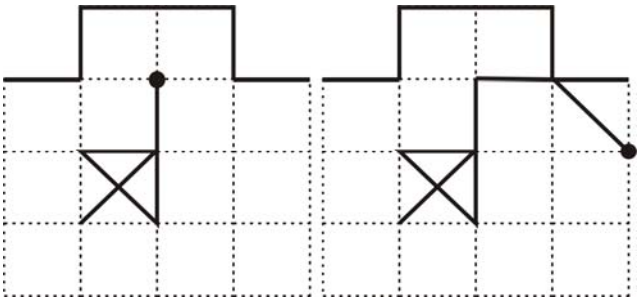


Figure 3.9 Sample almost goal situation

With the knowledge of most important game rules, it was a time to start develop a program using described platform which is Java Micro Edition.

Chapter 4

Inside the game - coding

At the beginning

After the IDE was ready and set up, with the rules of the game explained, everything was ready to start implementing described in Chapter 3 rules. But because of my very small knowledge about Java ME at the beginning, I have started from very simple programs which I found on the internet [2]. These were basic like “Hello World” program, going through Key event handling, finishing on very simple game. But all these lessons gave me very good understanding of how to program using J2ME language and enabled me to develop Mobile Soccer.

Program flow

The game flow was considered very deeply to create the game which is easy to navigate and be intuitive. This is by the occasion one of the requirements of the proper game. The diagram on Figure 4.1 presents how player can move between the screens. At this point it is important to remark that the classes responsible for the game flow are GameMainClass and Menu class which are described in more detail later in this chapter.

At first, when the game is launched, a splash screen (using SplashScreen class) is displayed which can be removed by pressing any button. After this, Menu class displays five options of the actions which can be performed. Single player and Two Player Games are implemented inside GameLogic class, whereas to display Help and About, TextBox class

(which is and extension to standard form class) launches the text form. Using Exit Game option from the menu causes program to launch exitMIDlet() method from GameMainClass.

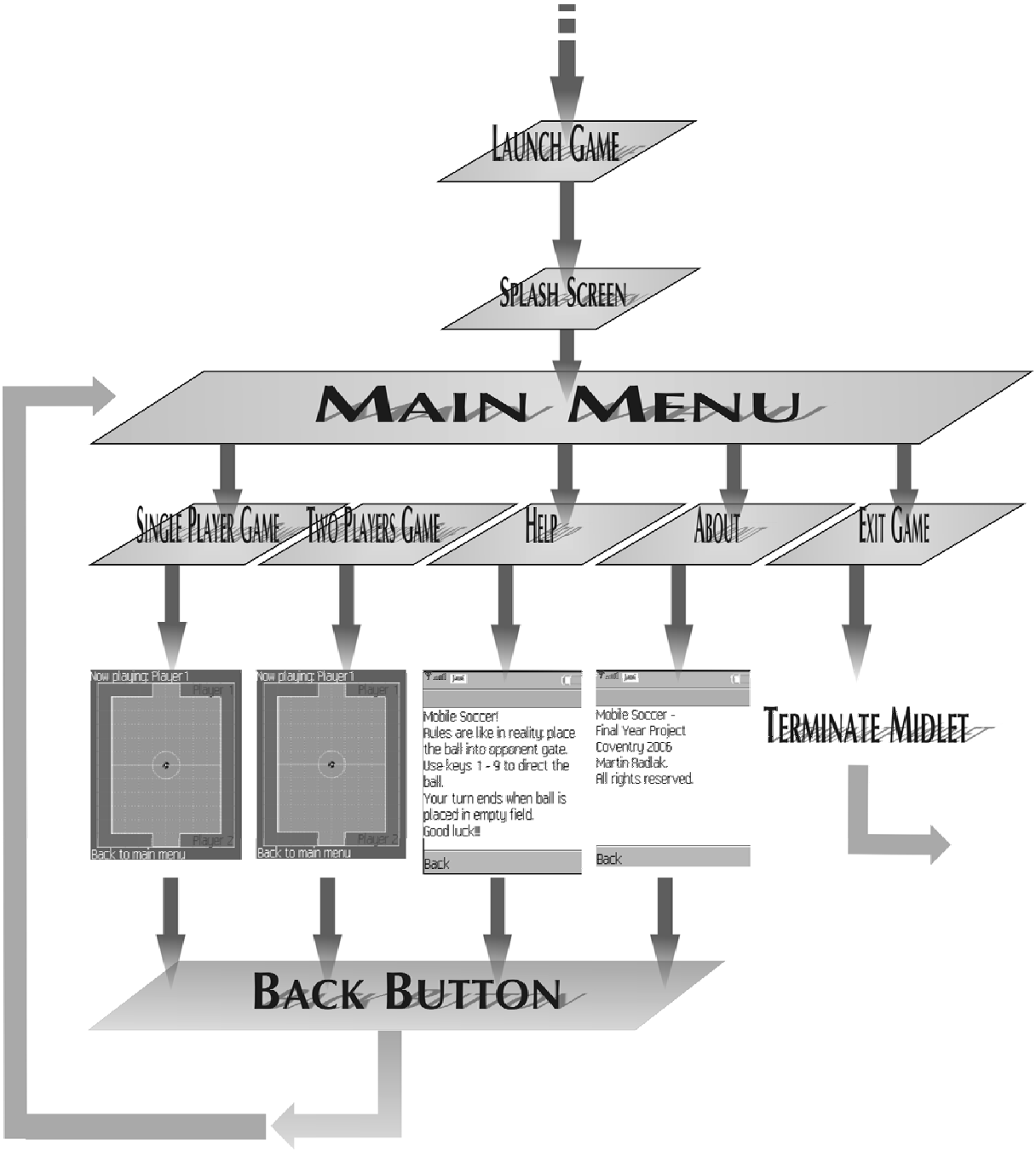


Figure 4.1 Program flow

From every single screen displayed after the splash screen, user can return to the main menu and chose different option. If for example playing Single Player Game, he decides to change to Two Players Game, it can be done by pressing back button to get back to the Main Menu and choose another type of game again. Only by choosing exit game option user will have to launch game from the beginning to get to the Main Menu

Program structure and GameLogic class

As it was described in detail in previous chapter, the core element of the game is a field. To transform it into a data structure The decision was made to use 3 dimensional array called gameArray which is of Boolean type. Its dimensions can be divided into: height - 13, width - 9, direction – 8 with a main task to store information about the moves that were performed. Array structure is shown in the Figure 4.2. Every move that is done, will be stored as a true value in gameArray, and is later used to check if the point of next move is available (availibilityCheck), or if after the next move, player needs to be changed (changePlayer method), or finally if the game is not finished (checkFinish method). These methods are also very important for the game so will be described shortly, specially how their tasks were implemented:

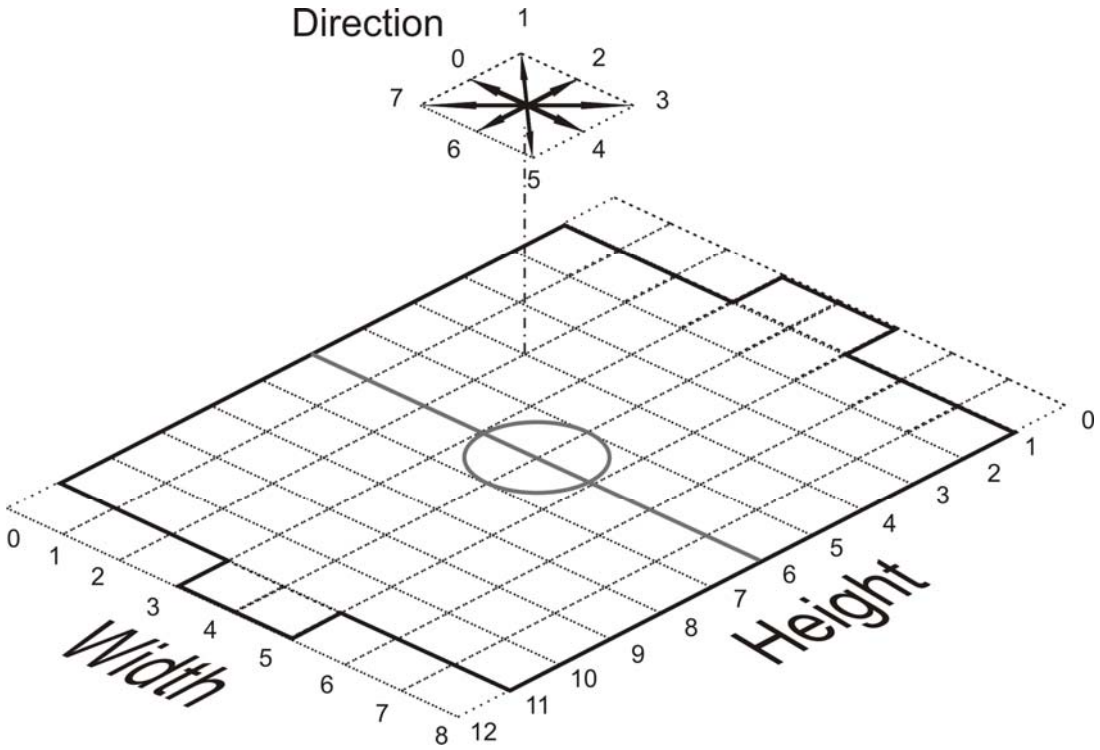


Figure 4.2 GameArray structure

private void availibilityCheck(int posX, int posY, int posXi, int posYi)

Listing 4.1 availibilityCheck method from GameLogic class

```
// check if the position pi is available
private void availibilityCheck(int posX, int posY, int posXi, int posYi)
{
    if (gameArray[posY][posX][dir]==false)
    {
        canGo = true;
        int vv = (dir+4)%8;
        gameArray[posYi][posXi][vv]=true;
        gameArray[posY][posX][dir]=true;
    }
    else canGo = false;
    //canGo = true;
}
```

This method makes heavy use of gameArray. It takes current position, next position and direction into calculations, and checks if the element gameArray[posY][posX][dir] = = false. If it is, then the move is available. It then assigns gameArray[posY][posX][dir] = true, but as the player can not go through the same line again - also the move in the opposite direction, from the future point needs to be set up as true – gameArray[posYi][posXi][oppDir]

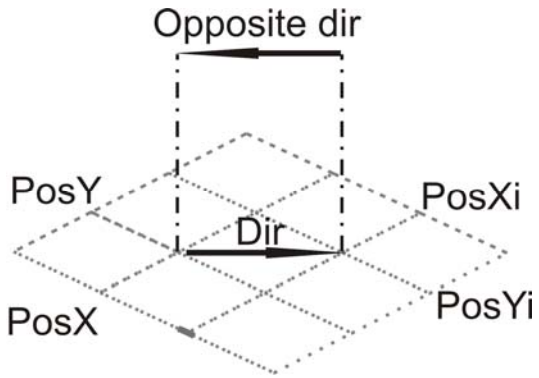


Figure 4.3 Way of disabling points that were already passed

Figure 4.3 explains the process of disabling points in gameArray that cannot be available again. Opposite direction is calculated using following formula

$$oppositeDir = (dir + 4) \bmod 8 \qquad \textbf{(4.1)}$$

private boolean changePlayer(int pX, int pY)

Listing 4.2. changePlayer method from GameLogic class

```
// check if the player change will be needed
// true - yes
// false - no
private boolean changePlayer(int pX, int pY)
{
    int n=0;

    for(int i=0; i<8; i++)
    {
        if (gameArray[pY][pX][i]==true) n++;
    }
    if (n>0) return false;
    else return true;
}
```

This method is responsible for changing the player which turn it currently is. It is done by checking the future position for every direction in gameArray. If any of them is true, that means that the player has already been in that point, so there is no need to change player and the method returns false value. Otherwise, returned value is true.

private boolean checkFinish(int X, int Y)

Listing 4.3 checkFinish method from GameLogic class

```
private boolean checkFinish(int X, int Y)
    {
        int n=0;
        boolean full = false;

        boolean gameFinished = false;
        for(int i=0; i<8; i++)
            {
               if (gameArray[Y][X][i]==true) n++;
            }
        if (n==8) full = true;

        if (((Y==0)||(Y==12))&&((X==3)||(X==4)||(X==5))) gameFinished = true; // if the player
        hit the ball into gate
        if (((Y!=0)&&(Y!=12)) && full == true)          // if there are no more empty paths
        {                                               // ....
            gameFinished = true;                        // indicate that the game has finished
            if (activePlayer == "Player1")              // and set active player to be the
        oponent
            {                                           // because oponent wins in this
        situtation
                activePlayer = "Player2";
            }
            else activePlayer = "Player1";
        }
        return gameFinished;
    }
```

Method which checks if the game is finished yet or not. There are few factors that decides whether the true (should finish) or false (not finished yet) values should be returned.

At first, the game is finished if the player has no more moves to chose from. This can be shown in a program, when for the specific point [X][Y] all the values of [dir] are set to be true. It is the situation when the opponent wins so is the string of activePlayer changed to opponent because it will be later displayed on the screen who has won.

Secondly, the game is finished, when the player directs ball into the goal. This is solved by checking the set of conditions, clearly seen in the Listing 4.3

These were the methods that are mostly important for the game. I think that there is no reason of showing the code of the other less important methods, as it can be examined in the appendix 1. To make understanding easier, the code is very well commented.

Remaining classes overview

Mobile Soccer consists of 6 classes, to handle the game flow:

- GameMainClass
- GameLogic
- Menu

- SplashScreen
- Directions
- TextForm

As the GameLogic was already described, I will present only remaining classes

GameMainClass

This is the introductory class for the game which contains methods that have to be implemented in every J2ME application. These are:

- `protected void startApp()`
- `protected void pauseApp()`
- `protected void destroyApp(boolean unconditional)`
- `public void exitMIDlet()`

These classes are to handle the most important states of the game and their names are clear enough to deduct which states are they responsible for.

I had to add another classes to handle other game states. These are shown below:

- `public void createNewGame()`
  this class is to switch display to Menu class;
- `public static GameMainClass getApp()`
  returns whether a game is running or not;
- `public void activateGameScreen()`
  switches display to show game screen;
- `public static void activateDisplayable(Displayable s)`
  activates display to show what was chosen before;

Menu class

The decision was made to make the process of creating menu simple. That is why in this class the use of Forms can be observed, that are available for programmers. It saved time preparing menu screen from nothing. The use of vector was needed, which contains menu choices and then assign actions to these options.

It is constructed of 3 methods:

- `launchMainMenu`
  prepares menu options and initializes vector storing them
- `prepareMenu`
  sets the title of menu and prepares it to draw to the screen

- `commandAction`

It is responsible for assigning specific action to the choice in the menu, initialize variables and launches the classes that are chosen

SplashScreen class

The aim of this class is to display a welcome screen with the name of the game and some picture. It is very common in every game to display a page at the beginning of the game to encourage customer.

To display a splash screen additional two methods were required: renderSplash and initializeResources. Because this class implements runnable, three methods are a must and so, they were already implemented: run, paint and keyPressed. To have a better view over these methods, it is recommend to examine source code which can be found at the end of this report.

Artificial Intelligence application

To make game fun and enjoyable it was decided to add some Artificial Intelligence to the program what fitted it with new challenges. The first goal was to develop a game, that two persons could play and have fun having a match between them two. But after that was achieved, I realized there is nobody who wish to help me testing the game. For this reason a decision was made to implement computer thinking, to make a game self sufficient. Because of lack of time and computation power AI implemented in the game is not very powerful, but good enough to provide entertainment.

Listing 4.4 CompMove method from GameLogic class

```
// Computes computer move
// input is a current position in Game Array
private Direction compMove(Direction pp)
{
    Vector bestMove = new Vector();

    for (int i=0; i<8; i++)
    {
        if ((gameArray[pp.pY][pp.pX][i]) == false)  // find all available moves from pp
    position
        {                                          // and store it in vector bestMove
           Direction temp = new Direction();
           switch (i)
           {
              case 0: {temp.pX = p.pX-1; temp.pY = p.pY;   temp.pdir =0;   }break;
              case 1: {temp.pX = p.pX-1; temp.pY = p.pY-1; temp.pdir = 1;   }break;
              case 2: {temp.pX = p.pX;   temp.pY = p.pY-1; temp.pdir = 2;   }break;
              case 3: {temp.pX = p.pX+1; temp.pY = p.pY-1; temp.pdir = 3;   }break;
              case 4: {temp.pX = p.pX+1; temp.pY = p.pY;   temp.pdir = 4;   }break;
              case 5: {temp.pX = p.pX+1; temp.pY = p.pY+1; temp.pdir = 5;   }break;
              case 6: {temp.pX = p.pX;   temp.pY = p.pY+1; temp.pdir = 6;   }break;
```

```
                case 7: {temp.pX = p.pX-1; temp.pY = p.pY+1; temp.pdir = 7;   }break;
            }
            bestMove.addElement(temp);
        }
    }
    Direction temp = new Direction();
    temp.setRR(12,14);

    // chose one of all moves
    for (int i=0;i<bestMove.size();i++)
    {
        Direction t = (Direction)bestMove.elementAt(i);
        if ((((t.pX > temp.pX)&&(t.pX<=4)) || ((t.pX < temp.pX)&&(t.pX>=4)))&&(t.pY <
    temp.pY)) temp = t;

            if( t.pY < temp.pY) temp = t;
    }
    // if did not find any turns, chose random one
    Random r = new Random();
    if ((temp.pX == 12) && (temp.pY == 14) && (bestMove.size()>0)) temp =
    (Direction)bestMove.elementAt(r.nextInt(bestMove.size()));
    return temp;
}
```

All the AI is implemented in GameLogic class in compMove method (Listing 4.4). The way of implementing it was firstly to realize which of the moves are available to be done. It has been implemented by making a loop, which checks move availability by testing value in gameArray around current point. If the move is possible to be done, its details are stored in bestMove vector. After the list is completed it is a time to choose one, which will bring computer player closer to player's 1 goal. Process of choosing the right option I was implemented using a set of conditions that may be described as follows: the move is better than previous one if it is closer to the middle of the field (horizontally) or / and if it is closer to player's 1 goal – closer to field end line (vertically). If none of above conditions can be fulfilled, one of the available moves is chosen randomly.

It may sound very trivial but this simple solution gave me quite intelligent computer player able to create situation forcing human player to think about the movement rather than easily passing it.

Chapter 5

Inside the game – playing and testing

Game's user end

To show a game flow, screenshots of the game will be presented. It will be printed in black and white colour but will contain all the necessary details.

The first action needed is launching a game after previously installing it in the phone. At first the splash screen is shown visible in following Figure 5.1. To have a better overview what is being described here it is suggested to have a view at Figure 4.1.

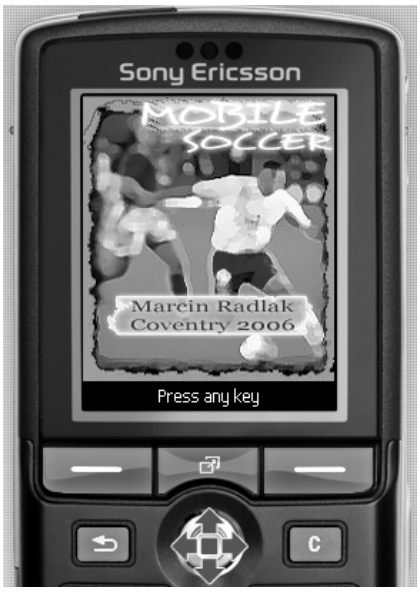


Figure 5.1 Splash screen

Pressing any key opens the Game Menu (Figure 5.2).

Figure 5.2 Main Menu

Choices shown are::

- "Single Player Game" – this is the type of Human – Computer game. It is the game we can play for pleasure and which does not need any other person
- "Two Players Game" – this is type of Human – Human game. To play we need to find a person who wish to play a match with us or we can train ourselves without any opponent
- "Help" – description and tips how to play
- "About" – this position on the menu is to show info about the author
- "Exit Game" – completely exits the game

Single Player Game

Choosing "Single Player Game" the field is initialized and finally displayed.

Figure 5.3 Field screen shoot

Figure 5.3 shows a screen shot of the initialized game. This is the key point now: if the game is “Single Player”, if we press any key that is not move we give a chance to computer to start. Keys that may be used to move are shown below in Figure 5.4:

Figure 5.4 Moves description

All keys, except 1-4 and 6-9, will cause computer to make its move first and it is not a fault of the game but its feature. This is very important to remember.

Another thing to remember is, that computer always makes his move very fast, so after we have finished ours it takes milliseconds until its turn is finished. Display in Figure 5.5 shows the state of game after computer has finished.

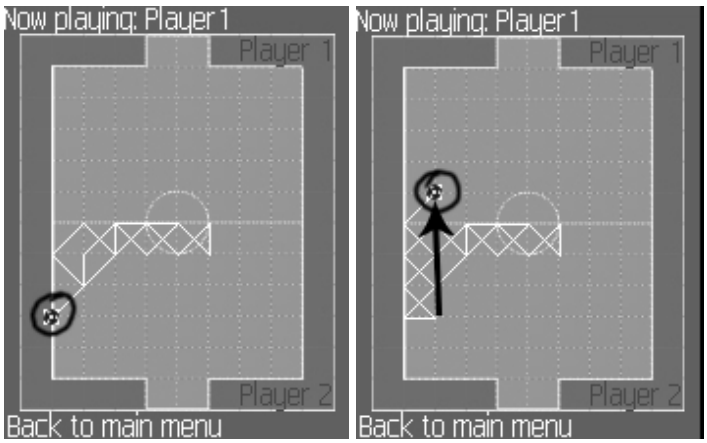


Figure 5.5 State of a game shows how rapid the computer’s move is done

Finally if the game is over, there is a message displayed saying that game is over and which player won.

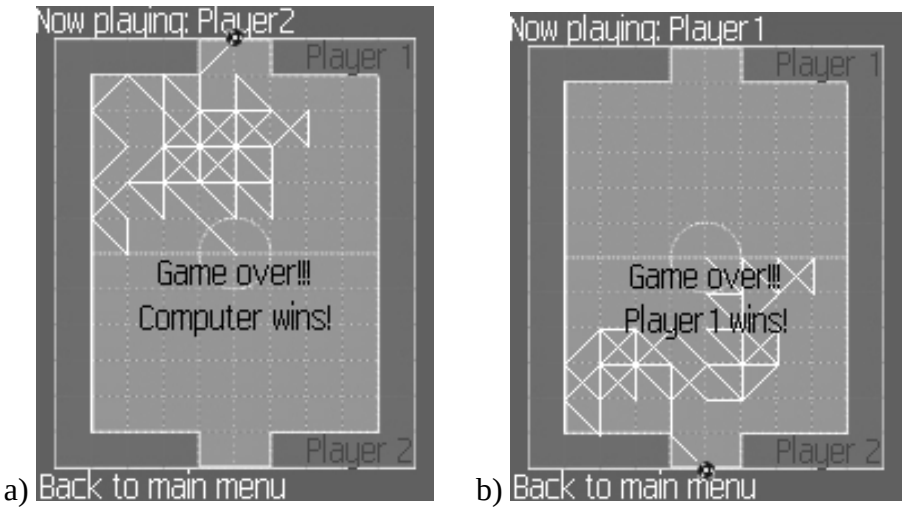


Figure 5.6 Two possibilities of finishing game: a) computer wins, b) human wins

To quit to main menu, pressing either back button or function button is required (Figure 5.4). This will activate the Main Menu.

Two Players Game

This type of game allows two people to play against them. This may be enjoyable time killer for two friends or a way to make tournament between larger number of individuals.

Game screen indicates (top left) which player's turn it currently is (Figure 5.7).

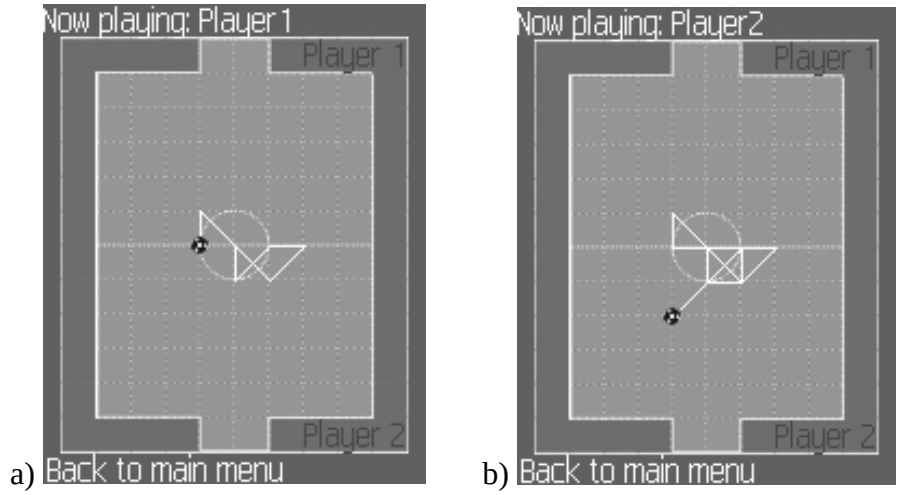


Figure 5.7 Game screen for Two Players Game: a) Player's 1 turn, b) Player's 2 turn

All the rules and screens are almost the same as in Single Player game so will not be described once again. The only difference is the end of game, where message displays which player has won. Instead of "Computer wins" it displays "Player .. wins".

Other Main Menu choices

Three other Main Menu choices (Figure 5.2) are not directly to play. They are done just for info about the game rules and about the author. Exit game option enables to close the game from application level.

Help contains the following text and is to explain basics of the game:

"Mobile Soccer!

Rules are like in reality: place the ball into opponent goal.

Use keys 1 - 9 to direct the ball

Your turn ends when ball is placed in empty field.

Good luck!!!"

About contains the following text and is to shows information about the game author:

Mobile Soccer –

Final Year Project

Coventry 2006

Martin Radlak

All rights reserved

To end game pressing Exit Game option from Main Menu is required or pressing and holding cancel button.

The game flow is not very complicated what makes it easy to navigate and intuitive. It is designed to be similar to the generally set standards of Graphical User Interface for games, and that is why it contains most common options to not confuse user.

Chapter 6

Discussion and further works

Discussion

As it is the end of the project I would like to present all the achievements. What I am mostly proud of is that I have created standalone computer program – game, which can bring everyone some fun and help to spend pleasant time.

The initial goal was to develop two player game with the rules described in Chapter 3. Reasonable amount of time has been spent to achieve the results that can be seen now. Significant visual differences may not be seen, comparing to the early stages, but there are huge changes in program code that were done since first presentation of game screen.

One of the bigger challenge was to refresh a screen after every key was pressed (move was done). The problem was to redraw the ball, without erasing lines showing previous turns. This was solved by storing those lines into a vector and redrawing them all with screen refresh. The ball was then printed on the top of everything. In similar way the problem of updating the string indicating which player's turn it currently is (top left corner) has been solved.

To make a game look and work like professional a menu together with splash screen have been added. Every proper game contain these two elements. Splash screen had to be displayed until the player pressed any key what was achieved. Game menu was prepared to give player choice of what type of game he wants to play, see help or information about game

and its author. Finally it enabled player to quit the game by choosing an option from inside it rather than using application handler to shut it down.

A lot of time developing the game logic has been spent. Rules are clearly explained in Chapter 3 so the task was to apply them. This process took huge amount of time because required translation from the language which people speak into machine one, so into the set of conditions only. Development of them resulted in proper game flow after long process of debugging.

Additionally, AI in the game was implemented, which made a single player (human-computer) game available and widened the audience of the game.

I did not use any help from anywhere implementing every single solution in GameLogic class. I also tried and made a lot of effort to make every piece of code very efficient and in my opinion I have achieved it.

Further works

I think I have done a lot for this project. But even that there are still many ideas which could be implemented, to make the game grow. I suppose that adding them all could lead to the very powerful mobile phone game and with a bit of marketing, could be well sellable. In this chapter the most important improvements will be shown, as well as the hints for making game much more attractive to the public. Unfortunately, I could not implement them because of lack of time for the project, but I will keep described improvements in mind, so if there will be some more time, they may possibly be finished. In next few lines I will describe in detail what are the next improvements needed.

Portability

Game has been developed with portability in mind, but my resources were strictly limited only to few of my friends Java enabled mobile phones and software emulators. It is a must to test and make necessary error corrections to avoid customers disappointment. It will also be necessary to make a list of mobile phones, on which the game can be played.

Artificial Intelligence improvements

Intelligence of the game is not the worst, but still require many improvements. Algorithm of path finding which I have developed myself, is limited to choosing the best solution at the moment of movement and does not include further prediction, what could make a game being bigger challenge and more enjoyable if it was implemented. Also

introduction of difficulty levels would be an interesting idea which could widen the players community, because more people could find a level of challenge that suits them.

Expanding connectivity to Bluetooth, IRDA and GPRS technologies

It is available to have single or two player game. But when it comes to two players match it can only be done by swapping handset between each other. It would be very innovative to implement different means of connection enabling a game between two mobile devices. This will give players freedom while playing and incredibly improve experience of it. Bluetooth is, I suppose, the best technology to use, because it is free and do not require, like IRDA, unbreakable streamline between two handsets. Additionally, almost every current mobile phone is Bluetooth fitted.

Timer

Interesting feature which would count the time needed to make a movement (analogy to chess) This could fasten the game and make it more exciting.

Implementing all above would result in very powerful game which, I think, could even generate some profit.

CONCLUSIONS

The project I have just described is a huge success in my opinion. I have achieved all the goals described at the beginning of this report.

The main goal was to develop a game that is fully working and is enjoyable at the same time. I can say that it has been achieved, because of very positive feedback from my friends. I have also learned many aspects of project management and greatly improved my skills in java J2ME programming.

Regarding project achievements of its own, I have developed a program which can be run on any mobile phone fitted with Java platform and MIDP 2.0 profile. It is a fully working game, with choice of two players or single player mode. Single player game is possible because of implementation of Artificial Intelligence which I have designed myself.

I have also learned how to approach and solve problems arising during development process rapidly and very effective, focusing on the major goal rather than going deep inside.

This project is for me a great achievement and I am very proud of it, work I have put into and results I have achieved.

BIBLIOGRAPHY AND RESOURCES

[1] Martin. J. Wells, J2ME – Game Programming, Premier Press 2004

[2] Tutorial with basics of programming in J2ME from Hong Kong University of Science and Technology website: http://www.cs.ust.hk/cmb/projects/j2me

[3] White Paper, Sony Ericsson K750i, Sony Ericsson Mobile Communications AB, March 2005

[4] Sony Ericsson developer website: http://www.SonyEricsson.com/developer

[5] Java™ Platform, Micro Edition, for Sony Ericsson mobile phones, Sony Ericsson Mobile Communications AB, September 2005

[6] Net Beans developer website: http://www.netbeans.org

[7] James Keogh, J2ME, The Complete Reference, McGraw-Hill/Osborne, 2003

[8] Addison Wesley - Programming Wireless Devices With The Java 2 Platform, Micro Edition (j2Me), 2Nd Ed – 2003

[9] Wireless Java Developing with J2ME, Apress, 2003

[10] Wireless Programming in J2ME, Hungry Minds, 2002

APPENDIX 1

Source Code

Listing 0.1 GameLogic.java – core game methods.

```
import java.util.Random;
import java.util.Vector;
import javax.microedition.lcdui.game.Sprite;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.IOException;

/**
 * Class responsible for all the game logic.
 */
public class GameLogic extends Canvas implements CommandListener
{
    private Command cmExit;          // Exit midlet
    private String keyText = null;    // Key code text
    private GameMainClass midlet;
    private Image backgroundImg;     // background image
    private Sprite ball;             // ball sprite
    private Image ballImg;           // ball image
    private Direction p;             // current position
    private Direction pi;            // next position
    private Image osb;   // Off screen buffer image
    private Graphics osg;   // graphics context for the off screen buffer
    private String activePlayer;
    private boolean plFlag = false;
    private Direction d;
    private int keyCde;              // number of pressed key
    private int dir;          // direction to mark in a game matrix
                              // 0: -, 1: \, 2: |, 3: /, 4: -, 5: \, 6: |, 7: /
    private Vector lines;
    // field indicating the state of a game:
     // true - game is beeing restarted
     // false - game is due
    public static boolean restart=true;

    // Field indicating if the game is finished and the winner should be shown
    // true - game is finished
    // false - game is due
    public static boolean finished = false;

    // KeyCodeCanvas constructor. It is to initialize most important variables, loads images
        and sets the KeyCodeCanvas class
    // @param midlet main midlet, which is responsible for game flow
    public GameLogic(GameMainClass midlet)
    {
        this.midlet = midlet;
        // Create exit command & listen for events
        cmExit = new Command("Back to Menu", Command.BACK, 1);
        addCommand(cmExit);
        setCommandListener(this);
        this.setFullScreenMode(true);    // make a game dispalyed full screen

        // load images
        try
        {
            backgroundImg = Image.createImage("/field.png");
            ballImg = Image.createImage("/ball.png");
        } catch(IOException ioex) { System.err.println(ioex); }

        lines = new Vector();            // store all the moves done
        ball = new Sprite(ballImg);      // create ball sprite

        osb = Image.createImage(getWidth(), getHeight());   // create off screen buffer
        osg = osb.getGraphics();                            // create off screen graphic

    }

    // Reseting game settings for a new game
    public void gameRestart()
    {
        finished = false;                // game not finished yet
        plFlag = false;                         // first turn is not a computer
        p = new Direction(4,6);                 // set the ball in the middle of the field
        pi = new Direction(4,6);
        dir = 0;

        activePlayer = "Player1";        // first player is player 1

        lines.removeAllElements();       // clear all stored move lines
```

```
        lines.addElement(new Direction(this.atcX(p.pX),this.atcY(p.pY))); // save the begining
        point of a game in vector

        ball.defineReferencePixel(4,4);         // set refference pixel of a ball sprite in
        the middle of its picture
        ball.setRefPixelPosition(this.atcX(p.pX),this.atcY(p.pY));  // set the ball in the
        middle of the field

        createField();                  // create gameArray

        osg.setColor(0, 128, 64);       // color for the fir layer of background
        osg.fillRect(0, 0, getWidth(), getHeight()); // and draw it on the whole screen
        osg.drawImage(backgroundImg,this.atcX(p.pX),this.atcY(p.pY),Graphics.VCENTER|Graphics.
        HCENTER); // draw background image
        osg.setColor(128,64,0);             // set color for description strings
        osg.drawString("Player 1",120,14,Graphics.TOP | Graphics.LEFT);
        osg.drawString("Player 2",120,189,Graphics.TOP | Graphics.LEFT);
        osg.setColor(255,255,255);      // set color for active player string and back
        command
        osg.drawString("Now playing: Player1",0,-2, Graphics.TOP|Graphics.LEFT);
        osg.drawString("Back to main menu",2,getHeight()-15,Graphics.TOP|Graphics.LEFT);
        ball.paint(osg);                // paint the ball int the middle of the field

    }

    // method preparing background into off screen buffer
    private void renderBackground()
    {
        //osg.drawImage(backgroundImg,getWidth()/2,getHeight()/2,Graphics.VCENTER|Graphics.HCE
        NTER);
        osg.drawImage(backgroundImg,this.atcX(4),this.atcY(6),Graphics.VCENTER|Graphics.HCENTE
        R);
        osg.setColor(128,64,0);
        osg.drawString("Player 1",120,14,Graphics.TOP | Graphics.LEFT);
        osg.drawString("Player 2",120,189,Graphics.TOP | Graphics.LEFT);
        osg.setColor(255,255,255);

        // draws all the previously made lines
        for (int i=0;i<(lines.size()-1);i++)
        {                               // begining if the line is an element i, end of line
        is an element i+1 of lines vector
            Direction test = (Direction)lines.elementAt(i);
            Direction testi = (Direction)lines.elementAt(i+1);
            osg.drawLine(test.pX,test.pY,testi.pX,testi.pY);
        }
    }

    // checks how many more moves are available from the pp  position
    // returns number of left turns
    private int numOfFree(Direction pp)
    {
        int n=0;
        for (int i=0;i<8;i++)
        {
            if (gameArray[pp.pX][pp.pY][i]==true) n++;
        }
        return 8-n;
    }

    // Computes computer move
    // input is a current position in Game Array
    private Direction compMove(Direction pp)
    {
        Vector bestMove = new Vector();

        for (int i=0; i<8; i++)
        {
            if ((gameArray[pp.pY][pp.pX][i]) == false)  // find all available moves from pp
        position
            {                                       // and store it in vector bestMove
                Direction temp = new Direction();
                switch (i)
                {
                    case 0: {temp.pX = p.pX-1;       temp.pY = p.pY;           temp.pdir = 0;
        }break;
                    case 1: {temp.pX = p.pX-1;       temp.pY = p.pY-1;   temp.pdir =      1;
        }break;
                    case 2: {temp.pX = p.pX;  temp.pY = p.pY-1;   temp.pdir =      2;   }break;
                    case 3: {temp.pX = p.pX+1;       temp.pY = p.pY-1;   temp.pdir =      3;
        }break;
                    case 4: {temp.pX = p.pX+1;       temp.pY = p.pY;           temp.pdir =
                4;   }break;
                    case 5: {temp.pX = p.pX+1;       temp.pY = p.pY+1;   temp.pdir =      5;
        }break;
                    case 6: {temp.pX = p.pX;  temp.pY = p.pY+1;   temp.pdir =      6;   }break;
                    case 7: {temp.pX = p.pX-1;       temp.pY = p.pY+1;   temp.pdir =      7;
        }break;
                }
                bestMove.addElement(temp);
            }
        }
        Direction temp = new Direction();
        temp.setRR(12,14);

        // chose one of all moves
        for (int i=0;i<bestMove.size();i++)
        {
            Direction t = (Direction)bestMove.elementAt(i);
```

```
            if ((((t.pX > temp.pX)&&(t.pX<=4)) || ((t.pX < temp.pX)&&(t.pX>=4)))&&(t.pY <
        temp.pY)) temp = t;

                if( t.pY < temp.pY) temp = t;

        }

        // if did not find any turns, chose random one
        Random r = new Random();
        if ((temp.pX == 12) && (temp.pY == 14) && (bestMove.size()>0)) temp =
        (Direction)bestMove.elementAt(r.nextInt(bestMove.size()));
        return temp;
    }

    // check if the game at given point didn not get to the end
    private boolean checkFinish(int X, int Y)
    {
        int n=0;
        boolean full = false;

        boolean gameFinished = false;
         for(int i=0; i<8; i++)
            {
               if (gameArray[Y][X][i]==true) n++;
            }
         if (n==8) full = true;

        if (((Y==0)||(Y==12))&&((X==3)||(X==4)||(X==5))) gameFinished = true; // if the player
        hot the ball into gate
        if (((Y!=0)&&(Y!=12)) && full == true)          // if there are no more empty paths
        {                                       // ....
            gameFinished = true;                // indicate that the game has finished
            if (activePlayer == "Player1")              // and set active player to be the
        oponent
            {                                           // because oponent wins in this
        situtation
                activePlayer = "Player2";
            }
            else activePlayer = "Player1";
        }
        return gameFinished;
    }

    // prepare off screen buffer
    private void render()
    {
        if (Menu.single == false) plFlag = false;       // if it is two players game, disable
        computer move calculation
        renderBackground();                             // create background
        osg.setColor(0xFFFFFF);


        if ((plFlag == true) )                                  // if it is computer turn
        {
            pi = compMove(p);                           // calculate next position
            dir = pi.pdir;
                                                        // save direction of the movement
            if (changePlayer(pi.pX,pi.pY) == true) plFlag= false; // if in the pi point player
        chang will be done, switch off computer
        }


        if (changePlayer(pi.pX,pi.pY) == true )                 // if player change is needed
        {
            plFlag = false;
            if (activePlayer == "Player1")              // chose correct player
            {
               activePlayer = "Player2";
               if (Menu.single == true)
               {
                   plFlag = true;
               }
            }
            else activePlayer = "Player1";
            osg.setColor(0, 128, 64);
            osg.fillRect(0,0,176,13);
            osg.setColor(255,255,255);
            osg.drawString("Now playing: "+activePlayer,0 , -2, Graphics.TOP|Graphics.LEFT);
        }

        // check if move to chosen position is available
        if (gameArray[p.pY][p.pX][dir]==false && p != pi)
        {
            canGo = true;
            int vv = (dir+4)%8;                 // opposite direction
            gameArray[pi.pY][pi.pX][vv]=true;
            gameArray[p.pY][p.pX][dir]=true;
            lines.addElement(this.arrayToCanvas(pi.pX,pi.pY));

            osg.drawLine(atcX(p.pX),atcY(p.pY),atcX(pi.pX), atcY(pi.pY));

            ball.setRefPixelPosition(atcX(pi.pX), atcY(pi.pY));

            p.pX = pi.pX;
            p.pY = pi.pY;

        }
```

```
        ball.paint(osg);

        // check if the game needs to be restarted
        if (checkFinish(pi.pX,pi.pY) == true)
        {
            finished = true;
            osg.setColor(0,0,0);
            if ((Menu.single) && (activePlayer == "Player2")) activePlayer = "Computer";
            osg.drawString("Game
        over!!!",getWidth()/2,getHeight()/2,Graphics.TOP|Graphics.HCENTER);
            osg.drawString(activePlayer +"
        wins!",getWidth()/2,getHeight()/2+20,Graphics.TOP|Graphics.HCENTER);
        }

    }

    /**
     * Repaint method (protected) to update the game screen
     * @param g this parameter is where all the graphic is drawn to
     */
    protected void paint(Graphics g)
    {
        // if game is not restarted neither finished
        if (finished == false && restart == false)
        {
            render();
            g.drawImage(osb, 0, 0, Graphics.LEFT | Graphics.TOP);

            while ((plFlag == true))
            {
               render();
               g.drawImage(osb, 0, 0, Graphics.LEFT | Graphics.TOP);
            };
        }
        else
        {
            restart = false;
            g.drawImage(osb, 0, 0, Graphics.LEFT | Graphics.TOP);
        }
    }

    public void commandAction(Command c, Displayable d)
    {
        if (c == cmExit)
        midlet.createNewGame();
    }

    /**
     * Captures the code of pressed key
     * @param keyCode Key code
     */
    protected void keyPressed(int keyCode)
    {
        keyCde = keyCode;

        switch (keyCode)
        {
            case KEY_NUM1:{
               pi.pX = p.pX-1;
               pi.pY = p.pY-1;
               dir = 1;
            } break;
            case KEY_NUM2:{
               pi.pX = p.pX;
               pi.pY = p.pY-1;
               dir = 2;
            } break;
            case KEY_NUM3:{
               pi.pX = p.pX+1;
               pi.pY = p.pY-1;
               dir = 3;
            }break;
            case KEY_NUM4:{
               pi.pX = p.pX-1;
               pi.pY = p.pY;
               dir = 0;
            }break;
            case KEY_NUM5:{

            }break;
            case KEY_NUM6:{
               pi.pX = p.pX+1;
               pi.pY = p.pY;
               dir = 4;
            }break;
            case KEY_NUM7:{
               pi.pX = p.pX-1;
               pi.pY = p.pY+1;
               dir = 7;
            }break;
            case KEY_NUM8:{
               pi.pX = p.pX;
               pi.pY = p.pY+1;
               dir = 6;
            }break;
            case KEY_NUM9:{
               pi.pX = p.pX+1;
```

```
                pi.pY = p.pY+1;
                dir = 5;
            }break;
            case KEY_NUM0:{
            }
        }
            repaint();
    }
    private boolean canGo;

    // check if the position pi is available
    private void availibilityCheck(int posX, int posY, int posXi, int posYi)
    {
        if (gameArray[posY][posX][dir]==false)
        {
            canGo = true;
            int vv = (dir+4)%8;
            gameArray[posYi][posXi][vv]=true;
            gameArray[posY][posX][dir]=true;
        }
        else canGo = false;
        //canGo = true;
    }

    // check if the player change will be needed
    // true - yes
    // false - no
    private boolean changePlayer(int pX, int pY)
    {
        int n=0;

        for(int i=0; i<8; i++)
        {
            if (gameArray[pY][pX][i]==true) n++;
        }
        if (n>0) return false;
        else return true;
    }

    private boolean[][][] gameArray;

    // creates the game Array cwith all the restrictions
    private void createField()
    {
        gameArray = new boolean[13][9][8];
        //delete upper line
        for (int x=0;x<9;x++)
            for (int j=0;j<8;j++)
            {
                gameArray[0][x][j]=true;
                gameArray[12][x][j]=true;
            }

        for (int x=0;x<9;x++)
        {
         if(x!=4)
         {
            gameArray[1][x][0]=true;
            gameArray[1][x][1]=true;
            gameArray[1][x][2]=true;
            gameArray[1][x][3]=true;
            gameArray[1][x][4]=true;
         }
        }

        gameArray[1][3][3]=false;
        gameArray[1][3][4]=false;
        gameArray[1][5][0]=false;
        gameArray[1][5][1]=false;

      for (int x=0;x<9;x++)
      {
            if (x!=4)
            {
            gameArray[11][x][4]=true;
            gameArray[11][x][5]=true;
            gameArray[11][x][6]=true;
            gameArray[11][x][7]=true;
            gameArray[11][x][0]=true;
      }
      }
        gameArray[11][3][4]=false;
        gameArray[11][3][5]=false;
        gameArray[11][5][7]=false;
        gameArray[11][5][0]=false;

      for (int y=0;y<13;y++)
      {
         gameArray[y][0][6]=true;
         gameArray[y][0][7]=true;
         gameArray[y][0][0]=true;
         gameArray[y][0][1]=true;
         gameArray[y][0][2]=true;
      }
      for (int y=0;y<13;y++)
      {
         gameArray[y][8][2]=true;
         gameArray[y][8][3]=true;
```

```
            gameArray[y][8][4]=true;
            gameArray[y][8][5]=true;
            gameArray[y][8][6]=true;
        }
    }


    // calculate gameArray values into game screen coordinates (X axis - width)
    private int atcX(int X)
    {
        return 16*X + 24;
    }

    // calculate gameArray values into game screen coordinates (Y axis - height)
    private int atcY(int Y)
    {
        return 16*Y + 14;
    }

    // calculate game screen (canvas) coordinates into gameArray values (X, Y axis - width,
        height)
    private Direction canvasToArray(int X, int Y)
    {
        Direction temp = new Direction();
        temp.setRRX((X - 24)/16);
        temp.setRRY((Y - 14)/16);

        return temp;
    }

    // calculate gameArray values into game screen coordinates (X, Y axis - width, height)
    private Direction arrayToCanvas(int X, int Y)
    {
        Direction temp = new Direction();
        temp.setRRX(16*X + 24);
        temp.setRRY(16*Y + 14);
        return temp;
    }
}

class Direction
    {
        public int pX;
        public int pY;
        public int pdir;

        public Direction(int X, int Y)
        {
            pX = X;
            pY = Y;
        }

        public Direction(){}

        public void setRRX(int X)
        {
            pX = X;
        }
        public void setRRY(int Y)
        {
            pY = Y;
        }
        public void setRR(int X, int Y)
        {
            pX = X;
            pY = Y;
        }

        public int rrX() {return pX;}
        public int rrY() {return pY;}

    }
```

Listing 0.2 GameMainClass.java

```
import java.util.Vector;
import javax.microedition.lcdui.game.Sprite;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.IOException;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.game.LayerManager;
import javax.microedition.rms.*;


/**
 * Main class which is responsible for the game flow. It takes control over every subclass in
the midlet
 */
public class GameMainClass extends MIDlet
{
    private Display  display;       // The display
    private static GameMainClass theApp;
    private GameLogic gs;
    private static Displayable currentDisplay;
    private Menu dm;


   // private KeyCodeCanvas canvas;   // Canvas
    private SplashScreen canvas;
    /**
     * Main class constructor
     */
    public GameMainClass(){
        theApp = this;
        display = Display.getDisplay(this);
        canvas = new SplashScreen(this);
        gs = new GameLogic(this);
        dm = new Menu();
        currentDisplay = canvas;

    }

    protected void startApp(){
        activateDisplayable(currentDisplay);
    }

    protected void pauseApp(){
    }

    protected void destroyApp( boolean unconditional ){
    }

    public void exitMIDlet(){
        destroyApp(true);
        notifyDestroyed();
    }

    public void createNewGame()
    {
        currentDisplay = dm;
        activateDisplayable(currentDisplay);
    }


    public static GameMainClass getApp()
    {
        return theApp;
    }

    public void activateGameScreen()
    {
        gs.gameRestart();
        currentDisplay = gs;
        activateDisplayable(gs);
    }

    public static void activateDisplayable(Displayable s)
    {
        try
        {
            Display.getDisplay(getApp()).setCurrent(s);
        }
        catch (Exception e)
        {
            System.out.println("App exception: " + e);
            e.printStackTrace();
        }
    }

}
```

Listing 0.3 Menu.java

```
import javax.microedition.lcdui.List;
import java.util.Vector;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.Form;

public class Menu extends List implements CommandListener
{
    /**
     * if false - two player game
     * if true - one player game (with computer)
     */
    public static boolean single;

public void launchMainMenu()
{
    Vector items = new Vector();
    items.addElement("Single Player Game");
    items.addElement("Two Players Game");
    items.addElement("Help");
    items.addElement("About");
    items.addElement("Exit Game");

    String[] s = new String[items.size()];
    for (int i = 0; i < items.size(); i++)
        s[i] = (String) items.elementAt(i);

    prepareMenu("Main Menu", s, "Exit");
      //currentMenu = MAIN_MENU;
}
public Menu()
{
    // Use an explicit list as it's the best type to reflect a menu (a simple
    // of items you can select from).
    super("", IMPLICIT);
    launchMainMenu();
    setCommandListener(this);
}

public void commandAction(Command command, Displayable displayable)
{
    // If they hit the back key you need to step back from the current
    // menu to a prior one in the hierarchy.
     // Execute an item on one of the menus.
    if (command == List.SELECT_COMMAND)
    {
        String selected = getString(getSelectedIndex());
            if (selected.equals("Single Player Game"))
            {
               single = true;
               GameLogic.restart = true;
               GameLogic.finished = false;
                GameMainClass.getApp().activateGameScreen();
            }
            if (selected.equals("Two Players Game"))
            {
               single = false;
               GameLogic.restart = true;
               GameLogic.finished = false;
                GameMainClass.getApp().activateGameScreen();
            }
            if (selected.equals("Help"))
            {
                GameMainClass.activateDisplayable(
                        new TextForm(this,"Mobile Soccer! \nRules are like in reality: place
the ball into opponent gate.\n" +
                                "Use keys 1 - 9 to direct the ball.\n" +
                                "Your turn ends when ball is placed in empty field.\n" +
                             "Good luck!!!"));
            }
            if (selected.equals("About"))
            {
                GameMainClass.activateDisplayable(
                        new TextForm(this,"Mobile Soccer - \nFinal Year Project\nCoventry
2006\nMartin Radlak. \nAll rights reserved."));
            }
           if (selected.equals("Exit Game"))
            {
               GameMainClass.getApp().exitMIDlet();
            }
    }
}
public void prepareMenu(String title, String[] choices, String backCommandName)
{
    // Set the title of the menu.
    setTitle(title);
    // clear the current list & command
    int s = size();
    for (int i = 0; i < s; i++)
        delete(0);
    // add the new choices
    for (int i = 0; i < choices.length; i++)
        append(choices[i], null);
    }}
```

Listing 0.4 SplashScreen.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.IOException;
import javax.microedition.lcdui.Image;


public class SplashScreen extends Canvas implements Runnable
{
    private boolean running = true;
    private Image osb;   // Off screen buffer image
    private Graphics osg;   // graphics context for the off screen buffer
    private GameMainClass theMidlet;
    private int starFieldViewY;
    private Image title;
    private int fontHeight;
    private Font font;

    public SplashScreen(GameMainClass midlet)
    {
        theMidlet = midlet;
       this.setFullScreenMode(true);


       Thread t = new Thread(this);

        t.start();
       initResources();
    }

    private void initResources()
{
    // set up the screen
    osb = Image.createImage(getWidth(), getHeight());
    osg = osb.getGraphics();

    try {
            //manager = new LayerManager();
            title = Image.createImage("/splash.jpg");
    } catch(IOException ioex) { System.err.println(ioex); }

    font = Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN, Font.SIZE_SMALL);
    fontHeight = font.getHeight() + 2;

    osg.setFont(font);
}
private static final int MAX_CPS = 100;
private static final int MS_PER_FRAME = 1000 / MAX_CPS;

public void run()
{
    try
    {
        while (running)
        {
            // remember the starting time
            long cycleStartTime = System.currentTimeMillis();
            // do our work
            repaint();

            // sleep if we've finished our work early
            long timeSinceStart = (cycleStartTime - System.currentTimeMillis());
            if (timeSinceStart < MS_PER_FRAME)
            {
                try
                {
                    Thread.sleep(MS_PER_FRAME - timeSinceStart);
                }
                catch (java.lang.InterruptedException e)
                {
                }
            }

        }
       theMidlet.createNewGame();

        // fall back to the splash form at the end of our loop
//        theMidlet.activateMenu();
    }
    catch (Exception e)
    {
        System.out.println("App exception: " + e);
        e.printStackTrace();
    }

}
private void renderSplash()
    {
        // clear the background
        osg.setColor(0);
        osg.fillRect(0, 0, getWidth(), getHeight());

        osg.drawImage(title, getWidth() / 2, getHeight() / 2 - 10, Graphics.HCENTER |
Graphics.VCENTER);
```

```
        // draw press key text
        osg.setColor(0x00ffffff);
        osg.drawString("Press any key", getWidth() / 2, getHeight() - fontHeight * 1,
                       Graphics.HCENTER | Graphics.TOP);
    }

    protected void paint(Graphics graphics)
    {
        renderSplash();
        graphics.drawImage(osb, 0, 0, Graphics.LEFT | Graphics.TOP);
    }

    protected void keyPressed(int keyCode)
    {
        running = false;
    }
}
```

Listing 0.5 TextForm.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.IOException;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.game.LayerManager;
import javax.microedition.lcdui.Form;

public class TextForm extends Form implements CommandListener
{
    private Command back;
    private Screen home;
    private Image bg;

    public TextForm(Screen homeArg, String displayText)
    {
        // call the Form constructor
        super("");
       try{
       bg = Image.createImage("/boisko.png");
       }catch (IOException ioex){}
       home = homeArg;
        append(displayText);

        // add the exit
        back = new Command("Back", Command.BACK, 1);
        addCommand(back);
        setCommandListener(this);
    }

    public void commandAction(Command command, Displayable displayable)
    {
        if (command == back)
            GameMainClass.activateDisplayable(home);
    }
}
```

APPENDIX 2

Project brief, Gantt chart, Risk assessment form

Presented forms in this appendix are not signed, because I did not received originals back. These are of the same content, just to present that they were done and handed out on time.

Final Year Project Brief

Surname: *Radlak*

Forename: *Marcin*

Course: *European Engineering*

Project Supervisor: *Dr Q Zhou*

Project Title: *Java Programming for Mobile Phones*

Background

Java technology used in mobile devices is getting incredibly popular these days. J2ME SDK provides us a set of tools, that make developing process much easier. It is important to recognize and to develop mobile application using correct mobile device emulator. Getting familiar with these tools is very important to stay on track with all the news in the programmer world. Applications in this project will be developed using Sony Ericsson J2ME SDK connected with Eclipse IDE.

Aims & Scopes

The aim of the project is to get familiar and to learn how to work with environment used to develop applications for mobile devices. Whole developing process will include reading documentation, writing code, emulating, debugging testing and preparing documentation. Every step will show possible problems that may encounter, expected or often unexpected and how to solve them. The most important is to get confidence with the new technology which is Java Micro Edition.

Activities & Output

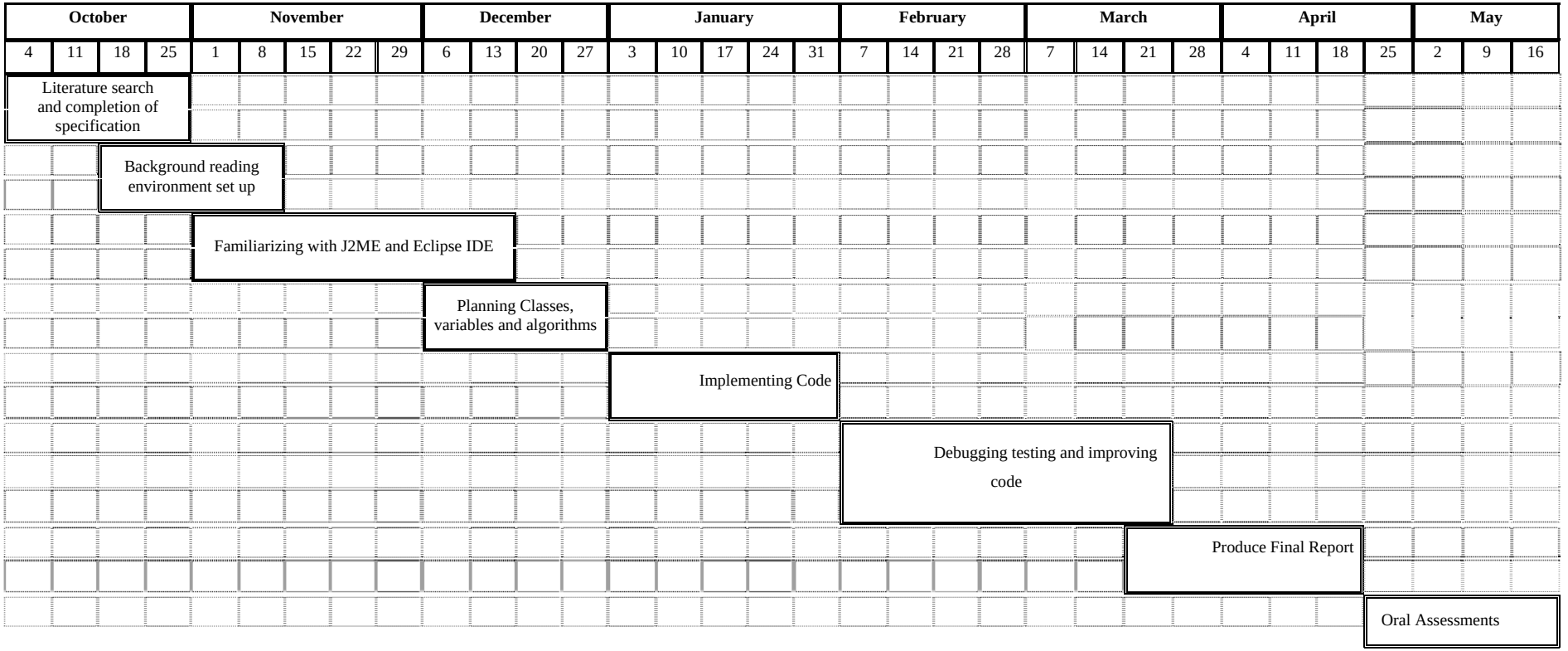
1. *Background reading*
2. *Preparing J2ME Platform and Eclipse IDE*
3. *Familiarisation with concepts and syntax of J2ME*
4. *Design of Classes, Methods, Variables and Algorithms necessary for application*
5. *Implementing code*
6. *Debugging*
7. *Testing and improvements*
8. *Writing formal report*

Student's signature *Supervisor's signature*

Date: *Date:*

Project Time Plan

Student : Marcin Radlak
Project Title : The title of your project
Supervisor : Your project supervisors name



Risk Assessment Form

Name of Student: Marcin Radlak

Supervisor Name: dr Q Zhou

Student Signature: …….

Supervisor Signature: ……

Location of Project: ……..

Date: 24 October 2005

PROCESS/ACTIVITY	HAZARDS	PERSONS AT RISK	ACTION TAKEN
Work with computer	Electric shock	Developer	Avoid broken wires
Software development	Eyes and back injuries	Developer	5 min brakes every 30 min and 15 min brake every 2 hours
Software development	Data loss	Developer	Frequent backups

Please return a copy of this form, signed and dated, to Dr. R.J. Rider, Q132, after the project specification has been agreed and before any practical work commences.

Further copies of this form may be obtained from Web location: **http://web1.eng.coventry.ac.uk/se_projects**

Interim Progress Report

Name: Marcin Radlak
Course: European Engineering Studies
Project name: Java Programming for Mobile Phones
Project Supervisor: Dr. Qin Zhou

Background

The aim of my project is to develop a game using J2ME technology. I am planning to extend playability of my game using Bluetooth technology to connect two mobile phones for multiplayer game. The program on its own is a football game, which will be available to play in different modes: player – computer, player – player (on the same mobile) and player – player (using two different mobile phones).

Current progress

I have started writing my project in October and now in the middle of December would like to share my achievements. I have started from preparing development environment. The first thing was installing an Eclipse software which is used to develop java applications. I spend a lot of time to get this software to work with Sony Ericsson development kit. Because I was not able to accomplish, I decided to change the development environment. My second choice was Net Beans Which is available on www.netbeans.org website. There are two versions I could chose from: stable version 4.1 and beta version 5.0. Beginning from version 4.1 I decided to choose version 5.0, because previous one was also not working. After setting the development environment, I started writing a code.
At first I copied simple programs from books and websites listed in Resources section. The one [2] showed me and let me understand how to write games a framework for the game. Using code samples I developed my own.
At the stage, I am now, I have developed the following components :

- game engine (using class …game.canvas)
- game screen (it is now displaying a bitmap, but I plan to make it much more portable, so the game screen will be painted by program
- game actions (actions that happens after pressing keys)
- game logic (boundaries and game rules)

I have had some problems with implementing the game logic but I solved it and everything works fine now. All the progress is going well and even better than I have planned

Resources:

1. Sun Java Website: java.sun.com
2. The Hong Kong University of Science and Technology: www.cs.ust.hk/emb/projects/j2me
3. NetBeans IDE: www.netbeans.org
4. Sony Ericsson Developer website: developer.sonyericsson.com
5. James Keogh, J2ME, The Complete Reference, McGraw-Hill/Osborne, 2003
6. Addison Wesley - Programming Wireless Devices With The Java 2 Platform, Micro Edition (j2Me), 2Nd Ed – 2003
7. Wireless Java Developing with J2ME, Apress, 2003
8. Wireless Programming in J2ME, Hungry Minds, 2002