Degree project

Difficulty of porting MVC Supervising Controller game from Windows OS to Android OS platform.
Abstract

There are hundreds of economic, strategy, sport, military and etc. games already in common use. As technology evolves, the number of platform on which this games run is increased. Each platform is different from another in interaction system with user, in technologies used for games to run on it or in capabilities of hardware. Sometimes game become popular and company wants it to be spread on other platforms or be created on multiple platforms. This is made to cover the bigger amount of auditory.

In order to have game on different platform it needs to be ported there. Porting process may be defined as transfer of game functionality to another platform. This might include rewriting code to another language and change game logic in order to accommodate to technologies used by this platform.

This process could be difficult and time consuming, because of platform- or technology-specific functions that might appear anywhere in the project. One of possible solutions is to make some of its parts be easier for porting. It means that the game project that needs to be ported should have specific architecture. This will make a clear separation between common processes (like game rules) and platform or language specifics.

This thesis work presents a game porting process from Windows OS to Android OS. The report will also present and discuss difference in efforts required for porting different parts of game project. Revealing of the moments that require attention and revision before porting.

Porting is considered to be partially successful, because of low performance level. The performance level was considered to be of no importance for our research. Thus it was not improved to the level of the original project. All data required to present difference between project parts successfully collected and analyzed. Model and Controller parts of the project were ported relatively easy while View part took most of total efforts spent on the whole project porting. The application has been tested on one physical device with appropriate results.

# Table of content

## 1 Introduction .................................................................................................................. 1
   1.1 Area of study overview .............................................................................................. 1
   1.2 Original project ........................................................................................................... 2
   1.3 Background .................................................................................................................. 2
   1.4 Problem ........................................................................................................................ 6
   1.5 Goals ............................................................................................................................ 6
   1.6 Restrictions and limitation ......................................................................................... 7
   1.7 The structure of the report ......................................................................................... 8

## 2 Method ............................................................................................................................ 9
   2.1 Method description ....................................................................................................... 9
   2.2 Internal threats to validity .......................................................................................... 10
   2.3 External threats to validity ......................................................................................... 11

## 3 Results and future works ............................................................................................ 12
   3.1 Results ......................................................................................................................... 12
   3.2 Discussion .................................................................................................................... 13
   3.3 Future works ............................................................................................................... 15

## Reference list .................................................................................................................. 16
1 Introduction

In this chapter we present the background that is required to know in order to understand the problem. We describe the goals, the area of study, the subject and object of study. We also describe the restrictions and limitations for this research.

1.1 Area of study overview

Nowadays games are widespread all over the world. According to Entertainment Software Association (2011) huge number of people spends their free time playing games. With such a high rate of interest in gaming, video games are springing up faster than mushrooms after a heavy rain. So it is obvious that the faster companies produce games the more people would play them. And of course the popularity of the company grows and it becomes well-known. It means it becomes more popular among other companies on the market. So, companies try to release game after game hoping the next one will be a hit. And if produced game is unforgettable and it catches people’s minds, company wants and tries to present it to as many players as it is possible. The way to do so is described in the work “Portability as an Aspect: Rethinking Modularity in Mobile Game Development” (Nui et al. 2011). Main idea is to transfer the game to a different platform or console or even release the game on several platforms simultaneously. This way this company will cover the biggest percentage and achieve success in the field of game production.

There are numerous devices which are used by people to play a wide range of games. Some examples are: consoles (Xbox, Nintendo wii, Sony Play station), platforms (personal computers, phones, Smartphone) and operating systems (Windows, Linux, Macintosh). They all are different and development of an application for console, for example, is different from its development for personal computer (Johannasen, 2011). The difference is in controls, hardware capabilities and whether this hardware could be changed or not. For example, consoles are closed box systems so the hardware is always the same and cannot be changed, so their capabilities remain static. At the same time personal computer can be modified each day with new hardware and it will make its capabilities vary between different models, etc. So, to increase the number of gamers the company needs its game to be ported with any or at least couple of devices mentioned above.

“Porting is a process of adapting software in an environment for which it was not originally written or intended to execute in.” (Technopedia, 2012). And devices on which game could be ported have different specifications of work, for example they can use different libraries to represent graphics, or can have different ways to interact with players. For example, joystick or keyboard can be used. This makes the porting process a hard, time- and resource-consuming work. The company wants a certain game to be ported as fast as possible, but it requires people with knowledge or experience, or time to produce them. So, as a result we might see that there is a strong need in making the process of porting faster and easier.
1.2 Original project
The object of the research is the original game project. It is the application that is made for personal computer with Windows operation system. The programming language used in development is C++. 3D graphic library responsible for object drawing, animation and graphical interface are DirectX.

Game-play is simple - player appears in 3D world and needs to find the way through the level from start to finish. In-game graphics depicts the world objects, like trees, ground, stones as a set of squares with textures and light (see Figure 1.1).

Following player controls are presented in the game: player can move forward, backward, to the left or to the right, he can jump, look around, shift to another side, restart the level and call the menu. Shift means change of current position to another. When player comes up to the edge of ground, stands near the tree or stone he can perform this action, which, basically, will make it look like the gravity change to the direction where this action was performed. For example, player after such action can walk over walls or ceiling. All actions performed by player are bound to a personal computer keyboard, except for look around - it is bound to a mouse movement.

Currently the game is in alpha version - only one level is playable, no animation for player movements, no sounds or music is presented.

1.3 Background
In this paper we examine the way of facilitation of porting computer games from one platform to another. This research is not the first one that touches the problems of porting from one platform to another. First of all, one of the examples of porting is a work of Bjorn Olsson, who did porting of application from IPhone operation system - iOS, to Android OS (Olsson, 2011). In that work he was trying to achieve the same
functionality with small changes in user interface and code style, appropriate for Java and Android and maintain appropriate architecture, which separates User interface from logic. As a result, Bjorn in his work showed that porting is successful. That means that porting application from different platforms to android is not only possible, but also these results may be similar to what original is capable for. And all that is without losing important features or abilities. His work was considered as basis for possibilities. And also it is a platform from which we should start trying new capabilities of Android devices. All this brings us to the idea of porting a personal computer game to Android, which includes more interesting challenges that require closer look.

Second work in this area that was chosen as relative and as a background of this research is “Portability Analysis in Mobile Gaming using J2ME” by Hari Menon (2006). Even though he did porting that did not include smart phones, his work has significant meaning. Hari showed that if the game is ported from a device with higher resources possession like higher memory or better graphic card to a device with lower resources possession the game has to be changed in the way it will be able to perform the same as original. That means the game should lose some of its functionality as well as good graphics and animation should be reduced and code refactored if needed.

If we compare conditions in which Hari Menon’s work presented the paragraph above, it is similar to our research, where original project runs on personal computer under Windows operation system and it needs to be ported to Smartphone with Android operation system. In fact, it is practically the same situation as described in Hari Menon’s work. To sum up: in current research original game project was made to run normally on the system that is more powerful than the system on which this game project should be ported. Taking into account this fact we modified research goals and work process beforehand: resource-consuming action controls of in-game player were eliminated from beginning; code style was changed from C++ language, which is a language of original project, to Java, which is language on which one can write applications for Android platform.

Second result of Hari Menon’s work is that each game should be created with possible change of domains in mind, which means each game should be developed portable in some way. This leads to a conclusion that specific approach is needed during the game project creation. The approach similar to the one that is presented in the work "Game logic portability" (BinSubaih et al. 2005) where the separation of game logic from graphics engine is presented and its portability discussed. Such approach should facilitate porting of project.

The approach called model-view-controller, MVC was used in current project (Lloyd and Cann, 2004). Mainly model-view-controller means a architectural pattern, which means its logic and functionality will be divided into three groups. Names of these groups form the name of the method: model, view and controller.
Model in the game project often contains coordinates of objects in the level, in-game rules and all the mathematics that hides behind it, not taking anything more complex than just positions and other specific characteristics that are a part of the world and do not touch anything that is relative to the system or drawing.

Controller is not connected with drawing either. This means all the drawing functionality should be excluded from it, if controller has any. In-game projects controller often stands for system, which means it is responsible to turn on and off in-game sounds or music. It is responsible for the game menu and it mainly reads and controls the state in which game resides. Also, in some specific cases, it controls whether game should response for inputs from gamer or not. The controller delegates requests to appropriate handler. These are means by which user interacts with the application. The controller is also responsible for the input to the model.

Finally, the view stands for the rest of a game project. View contains functionality that draws objects on the screen, holds data supporting drawing and parameters of objects, which do not connect it to the world it is drawn in. These parameters are color, position of points, of which 3D models consist relative to each other, textures, etc. Also sometimes view has a specific part which is responsible for receiving data from outside the system. It cans both be player inputs and changes in the environment, which might affect drawing style or resource possession. These are main responsibilities of view in game projects.

Many different model-view-controllers exist in the world nowadays. The one that was used during current game project development, which means the one that was used for the structure of the original project, is called “Supervising controller” (Fowler, 2006).
Figure 1.3 - Model-view-controller Supervising controller (presenter).

The idea of this model-view-controller is that we divide the project into three sections - model, view and controller (in this MVC controller is also called - presenter). Where: model contains not more than in-game rules (physical laws, objects collisions and mathematical calculations of position changes) and controller holds responsibilities of action distribution and represents the system, but it contains only top level and complex application logic (set course of program actions according to specific game type selected, etc). The rest - drawing, receiving inputs from the player, etc. is in the view. Basically, it creates a light controller and a heavy view.

The thing is that controller could be overfilled with functionality that is connected with receiving data from player and in some cases even small parts of representation. By moving all this to the view we leave controller only with plain functionality redirecting and game state check. All that should make it more draw- and platform-independent and make it a “porting neutral code”. The code requires just translation into a different language. And as far as in our game project model is already “porting neutral code”, as it consists mainly from mathematics, we get the view that needs to be reconsidered for porting. So, basically, this should decrease the code that needs to be changed and separate it from the code that just needs to be translated to a different language. In total, it should ease the porting, as it will decrease possible confusion and will practically show the person, who does porting, points on which he needs to pay attention to.
1.4 Problem

The problem that current research will try to solve sounds like: "How can model-view-controller Supervising controller influence on efforts that needs to be spent on different parts of game project during porting?".

Model-view-controller is a general software architecture pattern for interactive applications offering benefits for porting applications to different platforms, by encapsulating primarily user interface in a controller and view components. According to the pattern the model should therefore be straightforward to port, the controller should require more effort and the view should require most effort. We investigate if strict view encapsulation using the MVC-pattern is beneficial to porting computer games between radically different platforms.

Solving of this problem could show which part of this project will take greater part of efforts to be ported - model, controller or view. The assumption is that the hardest to port will be view. It is based on the particularity of supervising controller structure, which excludes most of platform or library dependent code from model and controller to view part. Basically, it should separate the code, which could be just translated to another language, from the part that needs to be rewritten.

It is a huge benefit for game producing companies to know the information about which part of the project will require the most effort to be ported. For example, project manager will know exactly that view part will be the hardest one during porting. So, this manager will allocate more people that have knowledge of technologies, which are used in the view, and less people that have knowledge in game-play programming. Or, as another example, the same manager needs to lead the game project through development with future possibility to port it on another platform. So, he plans hours and assign people to different tasks in this project. Using supervising controller and knowing the most difficult phase manager could allocate more time to game-play and mechanics, as logic should remain the same for every ported project. And less efforts and time on graphic library functions, animation etc., as most of it will be changed during porting.

Even if one person will need to port a game project such information would be beneficial. For example, knowing that view will be the most complex part to port, person, who performs porting, will spend precious time on reading about technologies that are used to represent graphic or do animation. Such knowledge will show him to what he should pay attention and will save time and efforts needed to port the game.

1.5 Goals

Three main goals were formed considering the problem that this research is trying to solve:

- investigate the possible difference in efforts that we apply to port different parts of model-view-controller of a game project;
- find and show possible places in project which could stall porting;
- suggest guidelines for improving portability in the different MVC-components.

The way of fulfilling current goals will be described in the chapter 2.
1.6 Restrictions and limitation

Major limitation for this research is that porting will be performed, tested and optimized for only one real phone - HTC Desire S.

OpenGL version should be the lowest one, which is 1.0 and android API should be 7, which makes it possible to run it on Android version 2.1 and higher. The reason for that is to cover greater part of Smart phones on the market (Google, 2012).

As you may see 1.5 and 1.6 versions cover a small part of the market (approximately 1% all together), but API difference would be great. So, there is no sense in trying to accommodate project under those versions. As 2.1 still covers 6% it was chosen as a start platform, as 2.1 project could also run on any versions of Android higher than current one.

Also we will not port some parts of the project, like:

- “Level Editor system”;
- “Shifting side player control”;
- “Entrance and exit points animation”;
- “3D featured main menu”.

The reason for that is stated in the background paragraph. When we were highlighting results of works in this area some of features were excluded during porting. It was done to lower the resource usage of the project on less resource possessive platform like Android, in comparison with personal computer. That was made after examination of all features of the project. Those features that were considered to be unimportant for current porting process were illuminated. Also no major performance improvement on the ported project will be performed, because current research is interested in straight porting and not modification of the project. The project should be considered as a finished unit. Modification is more about transferring project from high resource possessive system to low, and this study is about porting, but not improvement.
1.7 The structure of the report
The second part of this report contains information about the method according to which our research was performed. It has a research design where step by step explained which actions took place and in which order. Along with presentation and explanation of possible internal and external threads to the validity of the research results.

The third part of this report contains results received after the research was finished, discussion and future works. All the measured data presented in results. Discussion contains analysis of received data and conclusions about accomplished research, based on this analysis. Future works contains challenges that could be done after this research have been completed.
2 Method

In this chapter we present the approach, by which the research was performed. Then we show the steps that will be performed during this study. Also we discuss the internal and external threats to validity of the study.

2.1 Method description

As a way to achieve current goals we need to port successfully Windows OS game application made in C++ programming language with DirectX drawing libraries to the Android OS, which has Java programming language with openGL. Case study approach was chosen for this research. Inputs were chosen to be as much real as possible to simulate the most common situations in which any person who ports the game can find himself.

We came up with two approaches to perform this study. First was to give abstract points of complexity for the part of project after porting it. Second – to calculate hours that person invest in porting of each part. We have decided to chose second approach and mark day by day time spent by porting person performing porting from the beginning of work till its end. Also we mark pieces and parts that were done for each period of time. In this way measurement of hours spent for lines of code can be formed and time difference in porting of each part can be reviled, along with measurements of time spent on modification after that. Measuring the real number of hours would help to calculate the proportion more accurate than the abstract numbers of efforts.

Here are the steps we need to perform in this study in order to meet the set goals:

1. The environment will be set. All steps required to set up work environment for Android porting will be performed as Alvin Scudder (2011) explained in his work “Porting of an iPhone Application to Android”. Concerning the work with original project, installation of Visual studio according to the Microsoft manual is required. Check and write, internal path in settings according to the location of project files on a computer, if required.

2. Porting of the model component. On this stage no changes to functionality are supposed to be done, just translation of each function. Along with model functionality some specific mathematical functions used in model, but which are not a part of it, should be translated as well.

3. Porting of controller component. At this stage mostly the same straight translation, as with the model part, should be performed. As it has just a complex logic, that in most cases does not depend on platform or specific libraries, but some functions that communicate with view need to be taken into account, because view will change.

4. Porting of view component. First of all the sequence of connection and calls for drawing libraries needs to be performed here. Then all view functions used in controller should be set blank so, that the project can be run for the first time on the platform and it is possible to check if the basics of drawing done and all exceptions caught. Then function by function view needs to be filled with functions that are used in controller and those that use models data like position, along with object model files and level data files loading and parsing. Then, after all previous will be done, the sub goal is to make plain models appear on the screen. After that, game controls are the next stop. Player should receive abilities
to walk, look around, jump etc. After all controls that were planned are ported next steps are textures and light. To translate functionality of view for textures and light parsing from files and to add it to model rendering. After it is done the game in its abilities should be realized as it is planned.

5. Check if the ported project game-play and its idea correspond to the original one.

As the final step is done, the study is considered to be finished. All data will be collected in special research diary. Every day task and working hours will be explicitly written to it. During this study time will be measured with a regular clock, lines of code will be measured with the help of “Eclipse” program, which can show the number of lines in each file.

It has been decided by the original developers that regular meetings with them are to take place every week. Length of each meeting is about an hour. The goal of these meetings is elimination of intricacies, resolution of occurring problems and general advising on the process.

Also, a small Android application could be implemented before the study starts. The idea is to use simple objects like triangle and squares. Apply rotation and movement on screen touch or phone tilt, add colors and textures to those objects, etc. This application is considered as a prototype and is aimed to discover possibilities of OpenGL that is used on Android devices and to practice this graphic library functionality.

2.2 Internal threats to validity

First thing to announce is that all the results are based on human factor as research is based on study. This means further data presented should not be considered as a hundred percent result in each case of portation. Also, all data is based on skills of a person during simulation of a real porting process.

Talking about skills, the person, who performs porting, has no knowledge neither in language of the original project nor in drawing libraries of both original and porting platform usage. Basically, it will make process of porting more time consuming in some parts. This could affect number of hours that will be spent and lines of code in transferred project. To minimize the influence of this threat it was decided to create a prototype to receive knowledge and experience lacking for this research.

During current study porting person’s physical or biological condition changes might affect the process. But, as we will measure hours of work from the beginning till the end, if condition of person will get to a stage when it will affect the study, the work will be stopped until the condition will get back to appropriate level. It is acceptable as the study is not bound to exact date.

As for instrumental condition the same clock will be used to take all measurements. It will be constantly checked to be working and to be accurate with another three clocks work. This is the main condition of current research performance. So, the correctness of obtained data will be extremely close to reality.

But regular help provided by developers of original project can threaten validity. Giving answers for problems, which would have taken time to be solved by porting person himself, could lower number of hours that will be spent on porting. As well as exclusion of project parts, considered to be non-ported, from porting process. The less code needs to be ported the less hours would be spent on porting.
2.3 External threats to validity

As it was mentioned before, the porting person has no knowledge in both graphical libraries and language of original, only in language and platform on which to port. This could be an external validity threat. If we generalize the results to the level of game industry companies, such companies could have workers that either have knowledge both in technologies of original and platform on which to port, or no knowledge at all. So, ratio in their results could be different from our result’s ratio. As the number of hours spent on the view part, which actually holds functionality that works with mentioned libraries or platform, can be bigger or smaller than current measured hours.

Also, the project that is chosen to be ported is not a finished game. It has only one level, no animation movement of 3D models, no music or sound work, so it could be considered as alpha version - one of the stages of project completion, but not the release yet. It means that if any company will port finished game of approximately the same size, porting process of such parts as view and controller may take more hours, because of additional functionality.

The measurement variables that were chosen for this study are lines of code and hours, might be too specific, because companies could use days or number of files in its porting process calculation. But currently chosen variables could be easily translated into desired measurements, so it should not affect ratio in results and external validity.

As mentioned in subchapter 1.6, the project will be ported and tested only on one phone. This means that performing same porting work on any other model of phone could have different ratio in model and view, because such work might lead to unexpected situations during project run. These are wrong graphical representation, decreasing of performance quality or errors, which will require exclusion of functionality from model or view. Hours can be spent to fix it and it can have different lines of code in solution. That is going to give ratio that differs from our in view and model parts.

Also, game performance will not be improved. This will affect the hours needed for porting a model, as it contains numerous calculations that need to be performed in a loop and might take too much processor time. Company will be interested in achieving the same or similar level of performance on the ported device. This means that porting person will need to spend more hours to get the model run as required. It may give different ratio in results.
3 Results and future works

In this chapter we present the results of the research fulfilling the set goals. Then we analyze these results in discussion part and talk about the possible future works for this research.

3.1 Results

Porting of game project was done according to the method. Results could be seen on screenshots below.

![Ported game project launched on HTC Desire S](image)

Figure 3.1 - Ported game project launched on HTC Desire S

Measured hours that presented in results below are based on the pure time of translating code, searching for solutions, rethinking and remaking logic. Hours spent on learning new technologies, language specifics library specifics, etc. was not included, as it is based on personal skills and not important for our research.

At the end of the study the following results can be presented:
- Model took 1370 lines of code from initial 2350 lines of code in original for the period of time of 12 hours in total in 9 days. This was a pure straight translation of a code from C++ to Java. All parts of drawing, which is usually library specific, and inputs, which are usually platform specific, were taken out of the model due to architecture specifics.
- Controller took 440 lines of code from initial 660 lines of code in original for the period of time of 11 hours in total in 7 days. Approximately 50% of time was spent on pure translation of code, 15% - on creation of temporary barrier functions from view, so the controller could be normally checked on lexical errors, 35% of time was spent on fixing logical errors and trying to get the
project to run. As a result calling to calculations and then communication with blank view functions was established.

- View took 950 lines of code from initial 2060 lines of code in original for the period of time of 31 hour in total in 29 day. Approximately 65% of code was translated to another language, most of it had several changes, and other 35% needed to be changed on more than 50%. Or was not appropriate at all and needed to be accommodated for drawing library or platform specification.

During the study there were hours spent on learning: basics of C++, specifics of Java, work with openGL and DirectX, and Android specifics. Along with it, approximately from 8 to 11 hours, not in one piece, but separately, were spent on search, understanding and elimination of useless code, platform specific code in important parts of project, language specific style of coding fix, etc.

Following places that could stall the porting process were discovered during the research:

- DirectX Mathematical libraries were used. They are covered under its API, so there was a need to find substitutes to add them to the ported project. This brings us to a decision to include own mathematical libraries, that will be open to see its classes and functions in the project. It could save time on searching some specific function’s implementation and give the ability just to translate those.
- Usage of language and platform specific style of coding was found. As example, C++ function “sizeof”, close and constant work with memory allocation, usage of inline functions, overloading operators etc. Possible solution is to make functions in more simple coding style (the simpler the better and the more platforms can be covered).
- Data stored in C++ memory compatible way. It forces to remake functionality in case if language does not support storing data in same way.
- Some low level controls implemented or called in controller, in the meantime it should be done in the view. Transfer and creation of functions in view for those would fix situation.
- Unused resources like textures and model files have been found in the project. It is crucial for porting to check all dependencies, usage of files and remove everything that is not necessary before the porting starts.
- The code that never been used was met in the project. As a solution it could probably be deleted even from original before porting starts.

3.2 Discussion

According to the results we receive ratio that is presented below.
The following discussion will be based on the figure 3.2. First of all, the biggest part of code that our project has is in the model and was ported faster, as measured numbers above show. It was faster in regard to controller and view hours/lines and without additional work. Which means all the game rules and calculations, in other words “core of the game world” was exposed to a minimal changes, which leads to minimal number of possible errors and problems that could occur.

Second of all, controller is the smallest part of project due to chosen model-view-controller and as logical errors starts at this point it is much easier to fix them by not having number of small functions, as it is shown above. Like drawing that was transferred to view. By time and effort consumption it is on the second place.

Next but not less important is that view was changed a bit more from original in comparison to model and controller. But considering it more closely would reveal that logical functions behind view were almost identical after porting. Big part of changing lay down on drawing, where system called, in this particular case, for openGL, and also input system. Class that stands for catching inputs from player has also been modified to connect specific parts of platform. By time and effort consumption it is on the last place among others.

View took so much time, mostly because of a number of functions that contain calls of DirectX and view logic. Mixing project logic with covered by DirectX functions forced to a numerous searches for the solution in the Internet and further adaptation of such functions to receive the similar graphical representation as original. Those were decisive factors in porting view part, because, with each step of completion of this part, ported logic was revised according to different facts in new logic of the project and in OpenGL specifics. Overall, application required some code restructure: removal of big parts of unused code, creation of mathematical library classes in the project, etc. We drive to logical conclusion that project was not prepared to be ported. In addition to portability all highlighted occasions that were found during porting was not less important. Knowing of their existence and learning on others mistakes is important for porting, as it will help to fix everything in time and make process easier and faster. It shows possible preparation of project to porting process either on stage of development or on stage right before porting.

Performance modification should be applied to a part of the Model, which stands for calculation of collision with objects in the world. It was done without considering that this project will be ported and was completely appropriate to work on personal computer, which had enough resource to handle such logic. It is absolutely not suitable
to run on Android. For example, some filtering algorithms need to be applied to lower number of objects in the world to be checked for collision. Also separate thread for some calculation might be applied to lower the load of main thread and increase performance. The mechanics of Java and C++ memory allocation are different, which also brought to a performance problem. As the solution example needs to avoid continuous object creation in functions and stick to reuse of already created objects, etc.

As a result of the research we found out that performance of ported application is much lower than the performance of the original. It is connected to a resource possession of devices, so the project considered being playable, but much worse than original. Even so, the game project is still holding the same idea and mechanics, but with some small explained shortages. Graphics, colors and lights are different a bit from original, because of video drivers work difference of each device. But still, it considers being appropriate result, as it does not go in conflict neither with idea nor with the game-play of this project.

Overall difference between two projects - original and ported is insufficient, if not considering performance. Also, unfortunately, the study cannot be considered a hundred percent successful, because ratio of hours and lines of code in model, view and controller can change after improvement of performance.

### 3.3 Future works

As future works the first and main one could be to discover if model-view-controller, supervising controller, actually makes the game project easier to be ported. The reason to do that is to discover more general topic than current problem, which is “How do we facilitate porting process of the game project? Could model-view-controller Supervising controller help to facilitate porting process?”. As far as current research was not able to find an answer to that due to a different, more precise problem focus, it could be counted as future work.

Another work could be done in future is improvement of performance to similar level like original. As Android devices possesses different by the amount resources (memory, graphic card performance speed, etc) from personal computer, and it operates them in a different way it would not be a simple task to save functionality from one side and to make it run faster from another.

Also as future work might be done next: current project considered to be a small, casual game without Internet connection, etc, which means supervising controller could be tested in porting the big game with complex architecture logic, dependencies, probably multi-player abilities, etc.
Reference list


SCUDDER, A. (2011) Porting of an iPhone Application to Android. Independent thesis Basic level (degree of Bachelor), Linnaeus University.
