VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

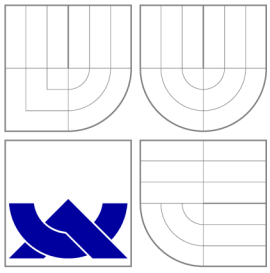
GAME FOR TEACHING VERY BASIC PROGRAMMING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

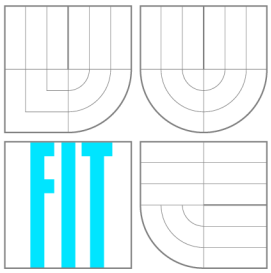
AUTOR PRÁCE
AUTHOR

MARTIN RONČKA

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HRA PRO VÝUKU ÚPLNÝCH ZÁKLADŮ PROGRAMOVÁNÍ

GAME FOR TEACHING VERY BASIC PROGRAMMING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN RONČKA

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2015

Abstrakt

Hlavním cílem této práce je vytvoření hry pro výuku úplných základů programování. První část této práce se zabývá studiem a analýzou současných her pro výuku programovaní a soudobých principů užívaných ve výukových hrách. Na toto navazuje návrh a implementace rozhraní pro vizuální programování v Unity3d a následná integrace tohoto rozhraní do jednoduché hry, která bude splňovat principy sepsané v první části této práce. Výsledek práce je poté vyhodnocen jak z hlediska technického tak uživatelského, s cílem zjistit efektivitu rozhraní pro vizuální programování a hry samotné jako nástroje pro představení programování.

Abstract

The main goal of this thesis is to create a game for teaching very basic programming. An analysis of current programming education games and education principles takes up the first part of this thesis. This is followed by a design and implementation of visual programming interface in Unity3d and later integration of this interface into a simple game supporting the foundings from the first part of the thesis. This thesis is then being evaluated from the technical and user perspective with the goal to analyze the effectiveness of both the visual programming interface and the game as a tool to introduce programming to non-coders.

Klíčová slova

hra, didaktická hra, vizuální programování, VPL, rozhraní, program, děti, výuka, multiplayer, soutěživost, scratch, codespells, loď, unity3d

Keywords

game, game-based learning, visual programming, VPL, interface, code, kids, education, multiplayer, competition, scratch, codespells, spaceship, unity3d

Citace

Martin Rončka: Game for Teaching Very Basic Programming, bakalářská práce, Brno, FIT VUT v Brně, 2015

Game for Teaching Very Basic Programming

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Adama Herouta, Ph.D.

.....
Martin Rončka
July 31, 2015

Poděkování

Chtěl bych zde poděkovat vedoucímu své práce panu doc. Ing. Adamu Heroutovi, Ph.D. za jeho inspirativní vedení a cenné konzultace při tvorbě této práce. Chtěl bych také poděkovat své rodině a své Lucince za jejich lásku a neustálou podporu.

© Martin Rončka, 2015.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Contents

Chapter 1

Introduction

The main goal of this thesis is to create a game for teaching very basic programming. The game should be entertaining all by itself but should also maintain its ability to entertain with the programming education aspects included.

In today's world, programming has becoming one of the most important skills for students and professionals to have. And programming is not just for programmers, it is has been widely used by designers and visual programming in some sort is widely common in the vast amount of modern tools in game development, 3d video productions, graphics tools and even audio tools. Having some programming basics is starting to be almost a minimal requirement and in some countries programming is already getting into the curriculum for elementary schools [3].

The following chapter 2 will cover multiple topics regarding existing programming games and education principles used in *Game-based learning (GBL)*. Here I will describe modern programming education games and several visual programming interfaces and try to analyze which of their key elements are an asset and where are their bottlenecks.

Chapter 3 describes the design of *visual programming language (VPL)*. Additionally this chapter contains all relevant information regarding the design of the game *UI (User Interface)*, *UX (User Experience)* and the design of the game itself.

In chapter 4 an implementation process of all previously mentioned elements will be described. This chapter will also cover up all surplus content that has been added to the game with the purpose of rounding off all the important aspects of the game.

Chapter 5 contains the test results and describe all relevant notes from the gameplay and suggestions for improvements by the players. Finally I will summarize what parts of the thesis were successful, where it lags and why.

Chapter 2

Research in Programming Education Games and Education Principles

In this chapter I will subsequently go through the existing programming education games (section 2.1) following with the description of some powerful visual programming tools in the section 2.2. I will summarize with simple introduction to visual programming in section 2.3. And finally, in the last section 2.4, I will go through educational principles that I have encountered during my research on Game-based learning to give this thesis some kind of leverage over other programming education games.

2.1 Popular Programming Education Games

Developments psychology research has shown the importance of play in learning [8]. In this subsection I will analyze the Games, from the perspective of playability and from the perspective of programming education incorporated. Games from the perspective of education are hierarchized into two primary categories.

Category 1 *Games which primary task is to entertain*

Category 2 *Games with primary task to educate, often called serious*

There is also a subcategory within the serious games:

Subcategory 2.1 *secondcategory Games which primary task is to entertain but but also contain educational value*

Subcategory 2.2 *secondcategory Games which primary task is to educate, but have incidental entertainment value*

Programming education games are mostly within the subcategory 2.2. The category 2.1 is less common with programming education games but in following examples there are games which incorporate very simple programming in engaging way even though they expose some of the programming basics (in form of VPL). These games are very often labeled as logical, but in fact they are programming games since they have VPL interpreter behind the scene. Games in this category are often more entertaining than the ones in the firsts category and thus more successful.

2.1.1 Karel

Very large portion of modern programming games derive from the old-school programming language Karel[1]. Karel had been the starting point of many programmers out there, and is still sometimes used at schools for introduction of the programming basics.

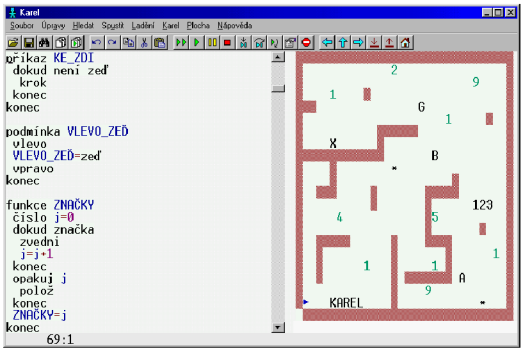


Figure 2.1: Czech Robot KAREL 4.2

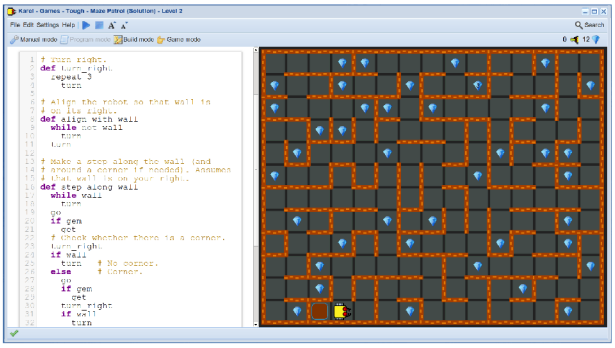


Figure 2.2: Modern version of Karel

The concept is based on a figure (robot) in a limited space, with limited set of commands and almost infinite number of tasks. The goal of the player is to write such procedures that will help him to solve given task, using the limited set of commands and subroutines the player can write himself.

Pros:

- Hands on code
- Writing down and maintaining own code in form of procedures
- Having to rely on your own code
- Simple Having to rely on your own code
- Large amount of extensions that focus on different programming paradigms

Cons:

- Out of date
- Visual representation might not be as appealing as in the old days

The main focus of Karel is to train procedural thinking, experience testing, debugging and use of subroutines [9]. These are one of the core elements of computer programming, and Karel is a great way to demonstrate that programming is not just something abstract code on a screen, but that it incorporates and deals with real devices and environment around us.

[1]The concept of Karel has become the basis of several other games such as CodeCombat, Kodable, Robozzle and many others.

2.1.2 Cargo-Bot

Cargo-bot is an iOS game that teaches the basics of programming by controlling a crane and stacks of boxes. Considering the limited amount of available commands and the need to create own procedures, the Cargo-Bot is much like Robot Karel.

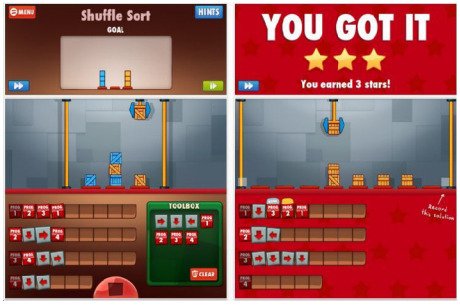


Figure 2.3: Stars for completing the level

Figure 2.4: Cargo-bot several examples for brain exercise

The main difference between the concept of Karel an Cargo-Bot is in the dramatic change of the visual side. Programming is executed through dragging simple command blocks to procedure slots which are then executed on play; level solution.

Cargo-Bot is a beautiful visual experience, the programming is simple but all the more fun and available for children. A great advantage of this game is it's simplicity and sophistication. With the little Cargo-Bot seems to offer, there are unreal amount of levels that can exercise your brain and give children some programming concept basics.

2.1.3 Lightbot

Lightbot is primarily a mobile game with its predecessor available online. Lihjtbot too is a programming puzzle game that allows to write procedures through similar interface like Cargo-Bot. Player is again limited with the commands he can execute but in every level is given a specific amount of procedures he can write and use to solve the puzzle.

Figure 2.5: Lightbot dialoges and character is made to attract children

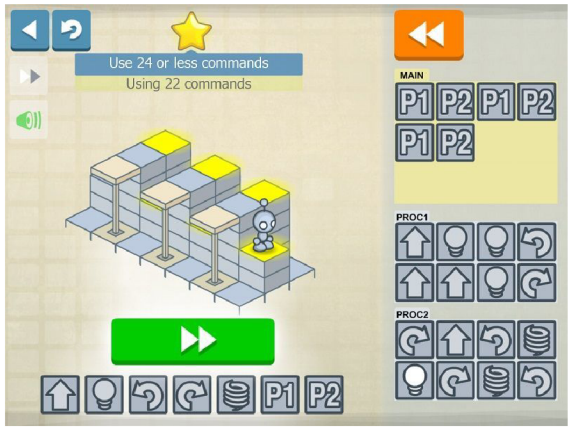


Figure 2.6: Lightbot motivates the players to write optimal procedures

Lightbots focus is to captivate children through it's graphics and simplicity. Character animation and dialogues are designed to attract children. Lightbot even offers version for small kids, Lightbot jr. (4+) with higher emphasis to the characters and guiding through the levels. Considering this, Lightbot seems as one of the best programming education games for children currently available on the market.

2.1.4 CodeSpells

CodeSpells is a modern game in development that has been successfully funded on Kickstarter. The game is being developed by a small team of Ph.D. students of computer science. The game itself is sort of an MMO-RPG that allows the player (sorcerer) to write down their own spells using VPL. This game uses specifically for the VPL an existing tool called Blockly (more at subsection 2.2.3).

Figure 2.7: Not your typical coding game

Figure 2.8: CodeSpells uses Blockly

Pros:

- Great and unique idea of how to use VPL
- Great visual aspect that can lure noncoders
- Freedom and fun with code

Cons:

- Development might turn out too difficult
- The game isn't focused for beginners yet

This game has a great potential and can become extremely popular among RPG players who code, but also can lure non-coders to the world of programming thanks to its visuals and freedom of creation.

2.1.5 CodinGame

Is the most notable coding games platform I have encountered. Providing large amount examples, using any given language, this platform is an ideal tool for development of programming education games using classical text based programming. Since this platform offers wide variety of tutorials it can be used to introduce text based programming to children.

Figure 2.9: CodinGame platform

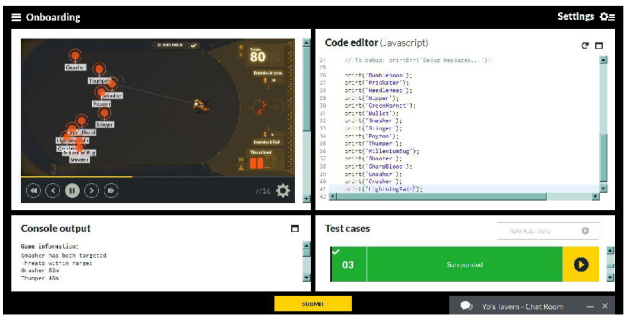


Figure 2.10: Running game with code editor

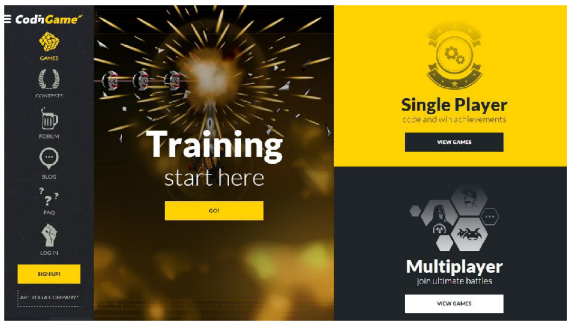


Figure 2.11: Training; Single/Multiplayer

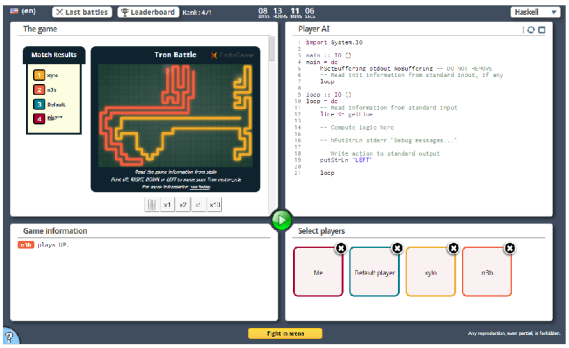


Figure 2.12: Multiplayer; Tron battle

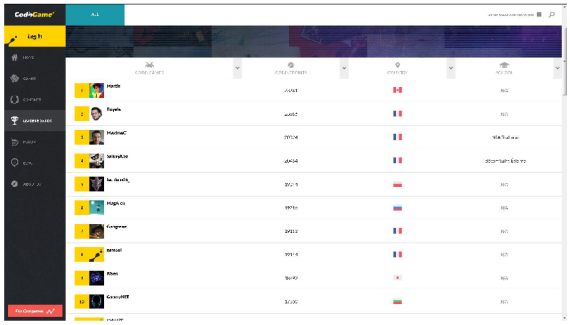


Figure 2.13: Leaderboards and contests are well designed

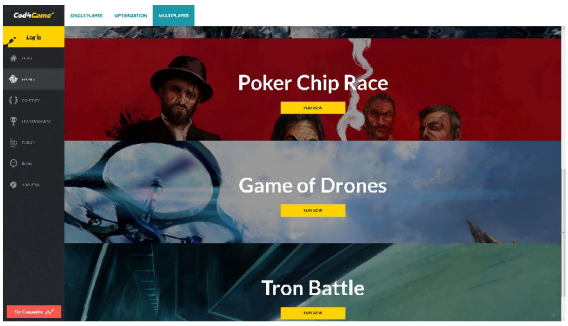


Figure 2.14: Wide range of games to choose from

But I have found these games austere and boring in comparison to the previously mentioned games. I believe that this platform can offer great games for programmers who want to start with a specific language or just to fool around with coding games.

Nevertheless, this platform is very well designed. It offers large amount of games to choose from, both singleplayer and multiplayer. Very often the multiplayer games consist of writing an AI script for specific task and then these AI's are being compared. This platform also offers leaderboards and contests to keep people interested.

Finding out this platform and realizing that there is not much to add I have decided to turn away from the text based programming game concept and focus more on the game development and introducing simplified version of programming using VPL.

2.2 Tools With Educative Programming Interface

Following sections contains the description of several very important programming education tools that stood at the beginning of the programming education or are at it's leading position now.

2.2.1 Etoys

Etoys is child-friendly object oriented visual programming language for use in education. It is based on Squeak which was built with the intention to develop an environment in which they could build an educational software that could be used-and even programmed-by nontechnical people, and by children [6]. Etoys is a product of that effort.

In development under the direction of Alan Kay[2] the Squeak Etoys has evolved into a media-rich authoring environment with a simple powerful scripted object model for many kinds of objects created by end-users. Etoys runs on multiple platforms, and is free and open source. It includes 2D and 3D graphics, images, text, presentations, videos and sound etc. It also allows to share desktops with other Etoy users in real-time, so many forms of immersive mentoring and play can be done over the Internet [7].

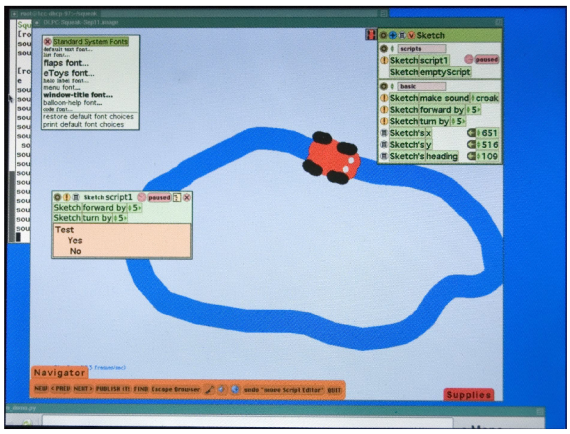


Figure 2.15: EToys is very graphically expressive

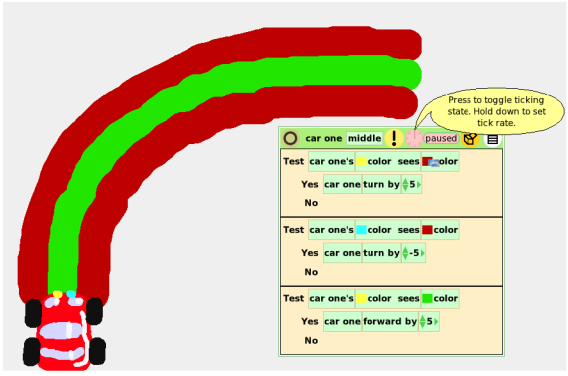


Figure 2.16: A car and a tile script which controls the car

Together with StarLogo[3] it has become one of the foundation stones of modern visual programming languages. These tools are a great way to introduce children to object oriented

[2] Great influencer of VPL languages and the initiator of Squeak and Etoys.

[3] StarLogo is an agent-based simulation language developed in MIT Media Lab with education purpose, predecessor of Etoys

programming. In this thesis, I would like to focus on the same goal. I believe that this is the right approach to teaching programming, by showing that there are real objects behind the code and it can destroy the barriers and make programming easier for children to understand.

2.2.2 Scratch

Scratch is another visual programming language developed in MIT Media Lab. Scratch has been highly influenced by the EToys, it is too very child-friendly VPL. Scratch is the most widely used modern VPL and can be accessed as a free desktop and online tool for creation of games and animations.

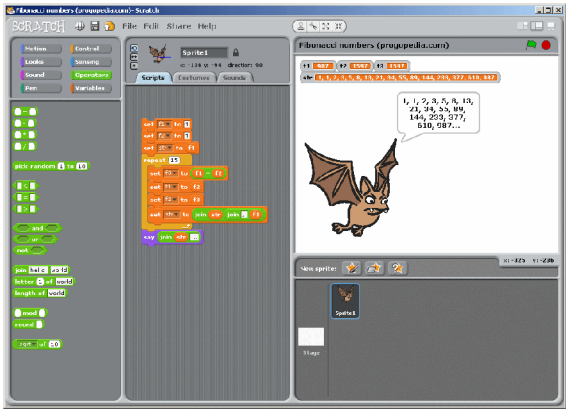


Figure 2.17: Scratch used to create simple animations

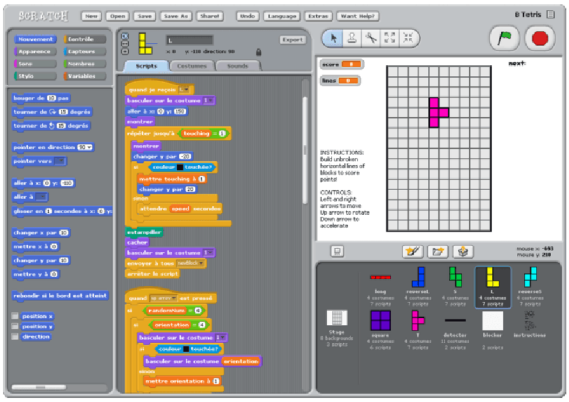


Figure 2.18: It can be used to create even complex games

Scratch brought the visual programming to a completely new level with simple and beautiful drag and drop interface. Allowing children to create simple games in a matter of minutes, Scratch is great progress from EToys. The visual interface of Scratch is something I want to learn from when developing my own VPL in this game.

2.2.3 Blockly

Blockly[4] is a library for building visual programming editors developed by Google. It is widely used among programming education platforms and even games, most notably by CodeSpells (subsection 2.1.4). Blockly offers example games and limited amount of examples on how to use their interface which speeds up development of visual programming editor. Blockly currently supports perfect interpretation of multiple programming languages including JavaScript, Python, PHP and Dart.

[4] https://code.google.com/p/blockly/

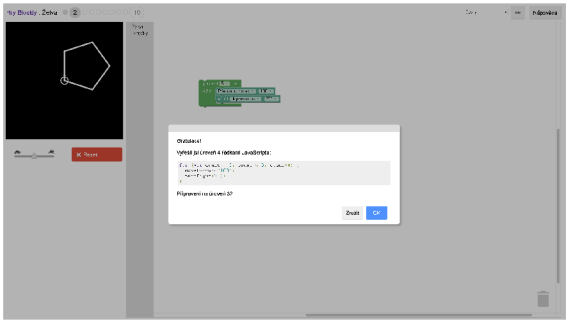


Figure 2.19: An example game, solving puzzle

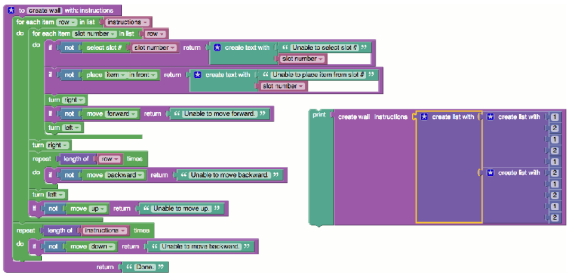


Figure 2.20: With Blockly you can create complex code

Finding out this library too late in the development process, I have decided not to use it but I would seriously consider the use of this library in future work regarding the education of programming or simply a development of visual programming interface.

2.2.4 Stencyl

Stencyl is a visual programming tool which originated with the MIT's Scratch that allows to quickly create games using its integrated VPL. But Stencyl is considered more of a game creation platform then education tool. More and more these visual programming tools are just simplification of programming and an abstraction of code even for non-coders among adults.

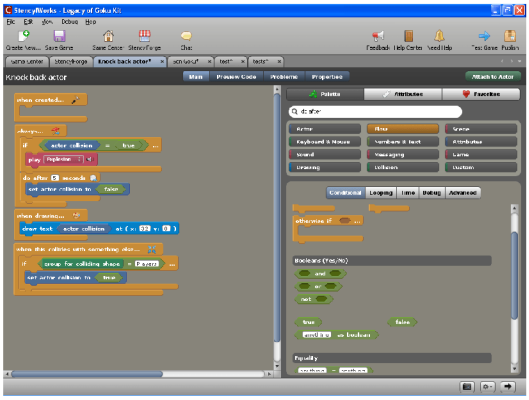


Figure 2.21: Stencyl offers wide range of blocks

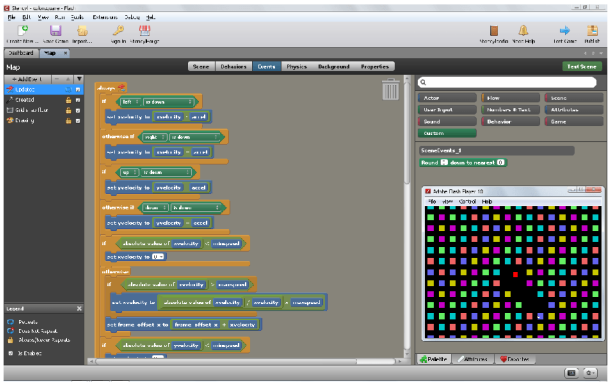


Figure 2.22: Stencyl games are portable accross multiple platforms

There are many other game creation tools with similar visual scripting features, including GameMaker, CraftStudio and even Unity offers a visual scripting extension in form of an asset called Playmaker.

2.3 Visual programming

In many of the mentioned games and tools there was been on very popular element. The programming itself is often not represented directly but rather through visualization representation. Generally this visual representation of code is called `Visual Programming`

Language (VPL) often also called visual scripting or drag and drop scripting. This concept has been popular since the beginning of computers but only recently (late 90s) it is getting into the focus of educators.

2.3.1 Development of visual languages

Visual programming is being vastly used among coders and non-coders to to get things regarding programming done without touching the code, usually at very high level. For example in Unity3D there is an asset that enables developers to do the scripting entirely using VPL (as shown in figure 2.24). There are several plugins that allow designers to use these blocks to program without writing any line of code.

As we've seen in section 2.1 and section 2.2, a lot of education games and tools use some sort of visual programming language to abstract the low level programming and make it easily accessible for kids and even for developers to speed up their work.

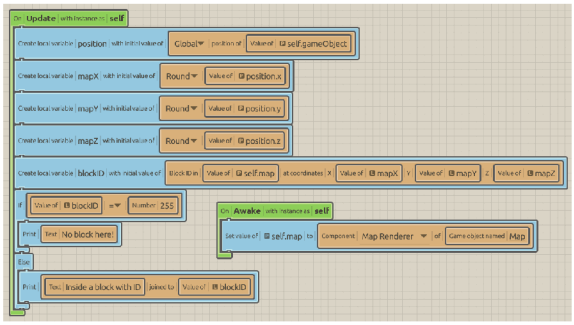


Figure 2.23: CraftStudio is a game-making platform

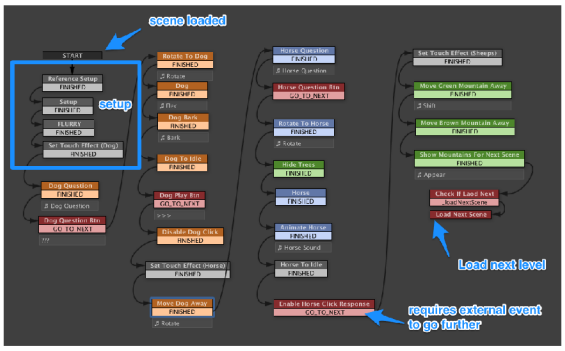


Figure 2.24: Even Unity allows visual scripting

Visual languages are very popular in multimedia, simulations or automation. Usually in form flowcharts representing finite state machines, system blocks etc. The main disadvantages might be that the users will get use to the high level abstraction and will not be interested in getting into low level coding. But it is the main objective to teach children a new set of reasoning and problem solving skills namely the creating (rather than following) the rules of a game [11] and this is very powerful tool to express this creativity..

Thus I have chosen to incorporate visual programming into the game instead of classical written code. There are several advantages to this decision. Such UI system would be applicable to touch devices which are not intended for long writing and watching visual feedback. This will allow the game to be ported to several more platforms. As Desktop games are hardly playable on Mobile devices without major UI changes, the backwards does not apply. If the interface will be designed in a good way, it will be playable on touch device as well as desktops.

2.4 Education Principles Used in Modern Games

A study conducted on educational games have come to a conclusion that students find education games engaging, but until they have provided clear goals with appropriate in-game feedback, students were not inclined to use the games for learning, or to pay much attention during or after the game to learning objectives. On the other hand, their results

indicate that tying game performance to learning objectives can improve student attitudes and engagement, which are two major components of learning [1].

Other studies show how to improve the amount of engagement when playing games and how to motivate children to play more. In this section I will describe such suggested improvements with according sources.

2.4.1 Reward System

One of the most important thing when dealing with education is the reward system. If the student does not have the feeling of satisfaction, the education process can become dull and boring. The same goes with games, if there is no progress and no reward, the enjoyment from game does not last.

Reward mechanisms in video games can enhance feelings of fun long before rewards are actually given-that is, rewards can create a sense of anticipation among players who know what is specifically required to earn them[5].

Achievements

Achievements give the player feeling of getting things done. Achievements are used in games to extend their longevity. There are mainly two types of achievements. Standard achievements are awarded to players for completing game, and going through the game to captivate the player's attention and give them the feeling of accomplishment. You can see an example of such achievement in the figure 2.27.

The second type of achievements are secret which are awarded only to players for accomplishing secret tasks. These achievements are meant to give the player the feeling of exclusiveness and rareness. Such achievements are sought by players who know the game and love it.

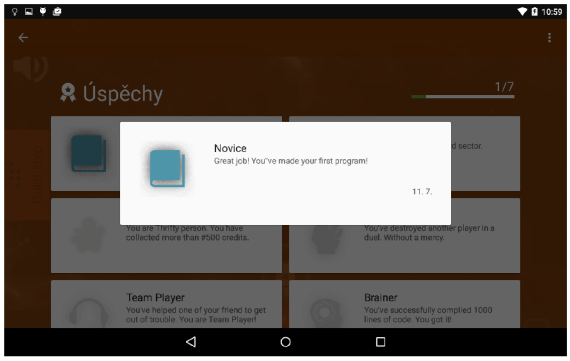


Figure 2.25: An example of achievement awarded in an Android game

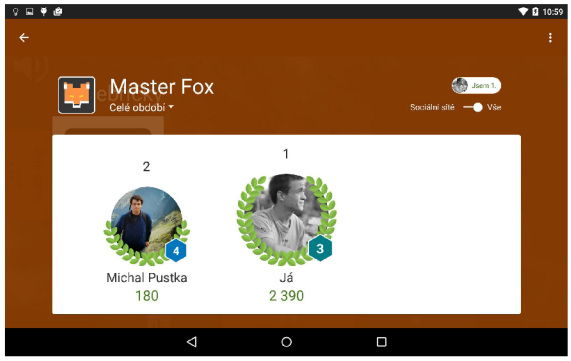


Figure 2.26: An example of a leaderboard used in an android game

Once the player feels comfortable he can seek achievements and play in a manner of fulfilling his recommended tasks. This plays role in Leaderboard. The Leaderboard consists only of score points, but these are awarded through both gameplay and achievements.

Leaderboards

Leaderboards are a way to compare how well players are doing. Leaderboards are usually structured in two main categories: the overall leaderboard and local leaderboard containing

friends who are connected to the player through social networks.

Developable Avatar

By providing developable avatar within the reward systems, it can create a sense of progress. Most of the games in which players control such avatar (mostly RPGs) use experience points for measurement. Player earns experience points during the gameplay for specific tasks or during for the gameplay itself, and levels up when specified goals are met. Experience points themselves create a sort of reward system, since they enhance avatars abilities [4].

Figure 2.27: An example of a Developable ship.

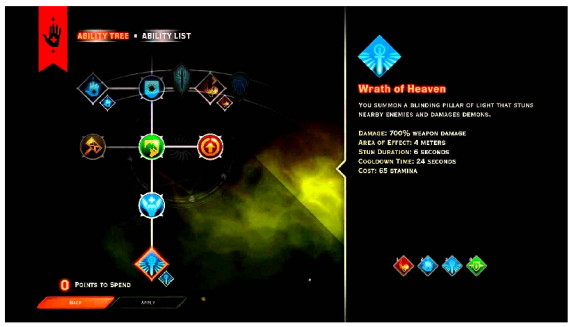


Figure 2.28: A skill tree example from a Multiplayer game

Rewards are often given in the form of new skills or increases in attributes such as strength, speed etc. Player levels affect gameplay in several ways, therefore almost all players are influenced by them. Players cannot precede with the game if they do not develop their avatars, and social-oriented players must gain sufficiently high levels in order to reach other players [5], thus thus motivating the player to play (learn).

2.4.2 Competition in Education

Both multiplayer and ranking are competition based elements. Competition is rooted deep within every one of us and by realizing it we have the power to offer children our own platforms to compete in. Naturally in our lives this is accomplished by sports, hobbies etc. This time, we are using competition to teach programming.

Several studies regarding this topic have been conducted and the the result is that a competition in an learning environment can be beneficial if it is designed following a number of principles, such as having a symbolic or little value prize, a short duration, and a goal clearly set into the (learning) process instead of into the results [2].

2.5 What Makes Games Good and Fun

Marc Prensky defines 6 structured elements for the purpose of game-based learning [10] that I would like to mention and hold upon:

1. Rules
2. Goals and Objectives
3. Outcomes & Feedback

4. Conflict/Competition/Challenge/Opposition
5. Interaction
6. Representation or Story

Also as part of my research I have been watching children play several games which I have presented to them and taking notes with the goal to understand what entertains them and keeps them playing. Writing down their reactions and asking them a few questions I have created a list of what I would like to focus on during the design and implementation of this game:

- Reasonable challenge
- Appearance - first impression is most notable
- Variation - The game needs to be dynamic either in appearance or in given challenges
- Comfort zone - Children should get pushed out of their comfort zone slowly

At the end of this thesis I would like to look back at how much does the game meet these set requirements. There are also many more attributes that I would like to achieve, such as novelty, intuitiveness and allure[12].

Chapter 3

Visual Programming Language, Interface and Game Design

In this chapter I will describe the design of VPL and the interface with respect to the matters discussed. Additionally this chapter contains all relevant information regarding the design of the game user interface, user experience and the design of the game itself. I would like to focus this thesis in the direction of a game from category 2.1 defined earlier in section 2.1.

3.1 Visual Programming Language Design

I have wanted the language to be as simple as possible and yet containing the very basic programming constructions, modern language stand on. I have decided to implement an interface for programming using blocks. Similar to Scratch, my goal was to implement a very simple interface with several code examples that could be thoroughly used in the game. Before the implementation, I have specified several implementation goals for the block programming interface.

I have designed a very simple language that supports basic programming constructions and that can be easy enough for a beginner to learn even without any previous experience. The language itself is not anything new, but rather focuses to be the minimal viable product in the scope of visual programming languages.

The language hierarchy is an implementation of design pattern `Composite` with ICommand being the `Component`, Statements containing other ICommands being `Composites` and other being the `Leaves`. This will allow easy link to the visual programming language.

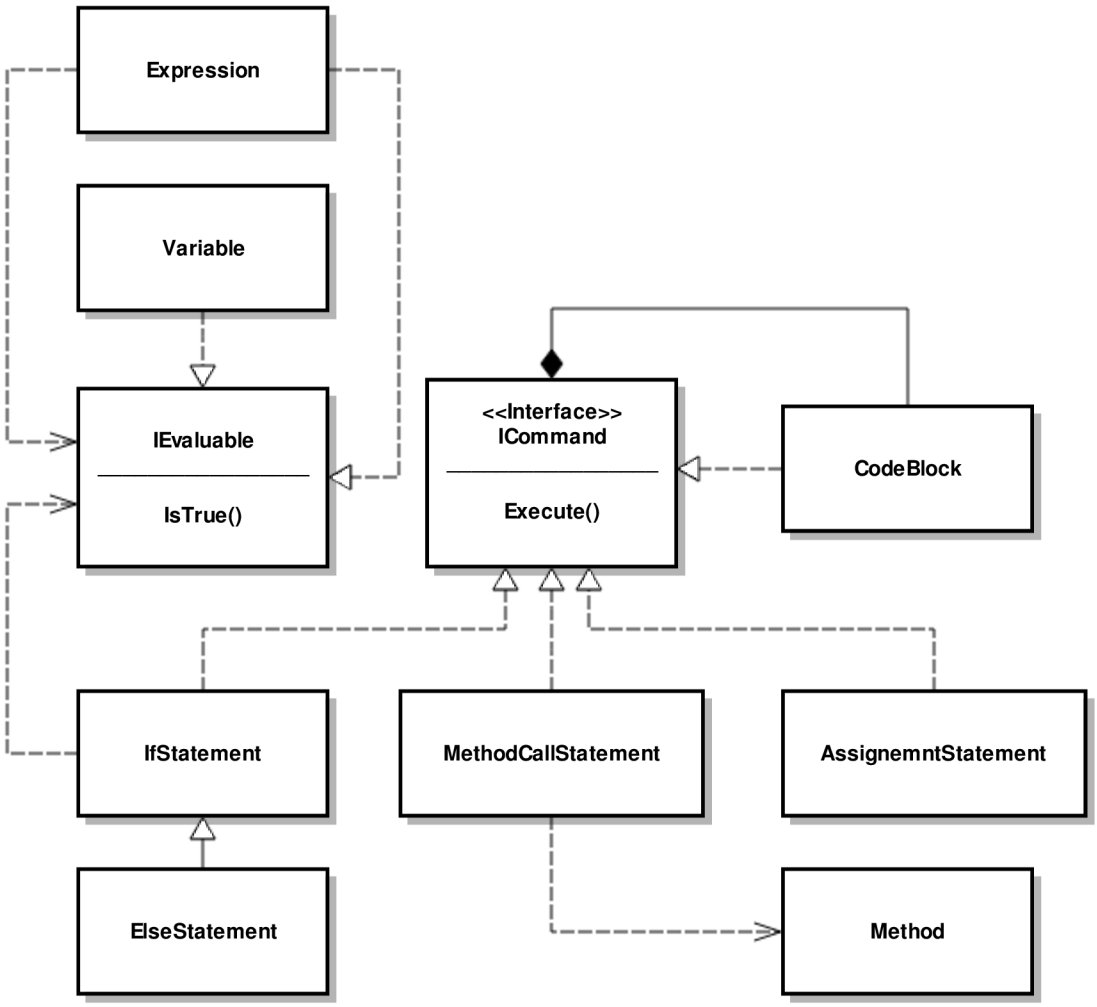


Figure 3.1: Object interpret class hierarchy

3.2 Nests

Nest are invisible grids that capture the `OnPointerEnter` event during the drag of any programming block. When a cursor or a touch enters the nest area, if an item is selected, a dummy instance is created and placed into the Nest. There are three main types of nest:

1. First - The dragged object will be positioned as the first object of a container (If blocks, loops, etc.)
2. Below - The dragged object will be positioned below the block, the nest is within (all blocks, simulates lines of code)
3. Instead - The dragged object will be positioned into the nest (parameters)

Each nest is placed within the programming block on block to the place that we want to capture for specific nest action. Nest is a main component of each Programming Block.

You can see colorized Nest in figure 3.2 where red colored box is a Below nest and Cyan colored box is a Nest of type First.

3.3 Indicators

When dragging a visual programming block an indicator is instanced once a nest is entered. This indicator will take a place inside the nest as the block would if placed inside. When outside of the nest, the indicator is partially transparent. The transparency should provide the feeling of nonexistence as you can see in figure 3.3.

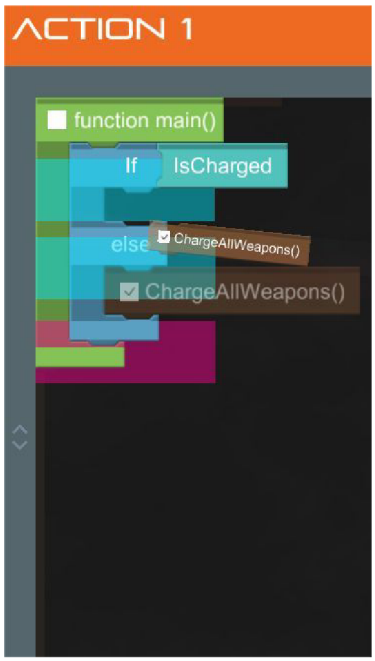


Figure 3.2: Colorized nests show the area of influence of blocks

Figure 3.3: Indicators to show the next or previous position

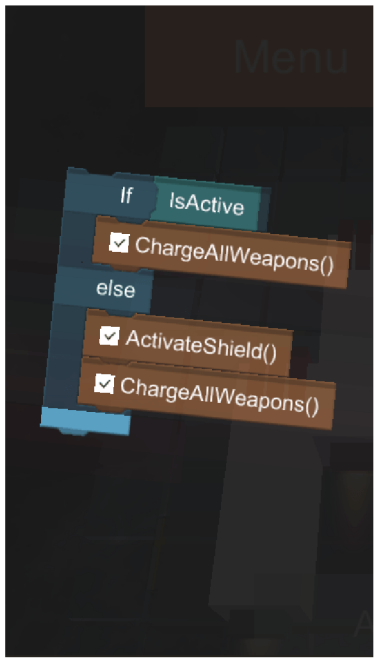


Figure 3.4: The tilt adds to the dynamics of the block interface

The block that is currently being dragged is tilted by 5 degrees. The tilt should invoke the feeling of dynamic action as it is. You can see the tilt in figure 3.4. Combining these elements I hope to achieve solid UX when working with these blocks.

3.4 Visual Programming Interface Design

I would like to encapsulate the programming into a manipulation with real world devices (in case of this game: ships system, weapons, shields etc.). The player will be The player will be able to see the an objects interface, all objects within it and methods the object contains. Then the player will be able to drag the blocks into a programmable procedure, effectively writing a program.

Figure 3.5: The ships systems can be seen at the right side panel

The player will be also allowed to use programming construction blocks for conditional statements, loops etc. The interface itself will also allow the player to undo and redo block dragging actions. I presume that this will come handy on touch based device where the dragging might not be the fastest action. When undoing a step, the block will either take its previous position in the procedure or will be moved to trash if it has been only just instantiated.

3.5 Game Design

The game design has been very iterative process since the very beginning. The main goal was to create game that would be in fact a game implementing the VPL interface as a bonus. Finally I have finished a very simple, yet powerful concept of a Space Shooter based action game using programming to mobilize the ships systems. The player is in the control of a ship with several programmable action slots. Using these slots, the player can access ships systems and program them to do anything the ship is capable of.

3.5.1 Achievements

I have designed a very simple way of awarding achievements to players.

First, the player will be awarded for completing the game story line. From first boss taken out to building the first blocks of ship. Then the achievements are directed more to the programming a multiplayer to guide the player towards the competition. For the first working program or Complex program without a bug etc.

Example of designed achievements:

Achievement title	Achievement description
Novice	Great job! You have made your first program!
Journeyman	You have reached the 3rd sector. Journeyman.
Thrifty	You are Thrifty person. You have collected more than #500 credits.
Ruthless	You have destroyed another player in a duel. Without a mercy.
Team player	You have helped one of your friend to get out of trouble. Thank you!
Hacker	You monster! You successfully sabotaged an enemy ship by hacking their systems.
Brainer	You have successfully complied 1000 lines of code. You got it!

3.5.2 Multiplayer

One of my considerations was to move to the mobile platform. And based on the analysis at chapter 2 I have decided to implement multiplayer element into the game game that will be touch based device specific.

Multiplayer requires either working networking or appropriate input management. With the concern for the scope of the thesis, I have decided to go with so called Hotseat multiplayer mode. Two players are engaged in the same game, competing or cooperating with each other. Cooperative play strengthens the bond between the players and the games.

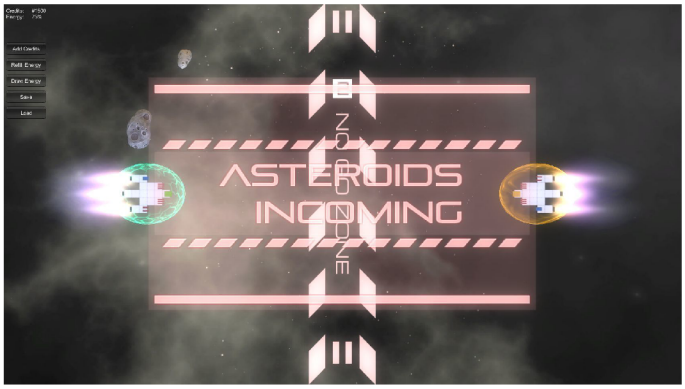


Figure 3.6: First multiplayer prototype

Once the player is engaged with the game and has competed with someone, he is motivated to move further, exploring the game, its capabilities and behaviours. This the ideal time to hit hard with some serious programming. That is, the basics. Both multiplayer and ranking are competition based elements. This is the key elements I hope to drive the players to play more.

3.5.3 Input Scenarios

When focusing on multiple platforms, the input management too is very important, what might work on desktop will not work on tablet device. This is why I have created simple finite state machine that will handle the input actions. Furthermore to easy the input handling, I will unify the touch and mouse input into a encapsulating class specifically for the need of this game. When playing on a desktop, the player ship can be additionally controlled using keyboard.

Chapter 4

Visual Programming Language, Interface and Game Implementation

In this chapter I will go through the selection of tools ideal for the game implementation (section 4.1). Then I will describe components used in the implementation of VPL in Unity (section 4.2) And then I will go through all key interfaces of the game (section 4.3).

4.1 Selection of Tools

In this section I will go through the the tools that I have tried at the beginning and that came into consideration for development of this game. The initial goal was to create web based game that would be playable in the browser thus being fully portable without the necessity of installation to the PC itself. These were the one of the core attributes that would define the game later in the process. But my approach to the development itself has changed. One of the possibilities would be creating a JavaScript based text editor supporting syntax highlight and setting this as an overlay to the Unity game in WebGL.

4.1.1 Three.js

Three.js is a JavaScript WebGL library. It allows high abstraction over the low level graphics API, but allows the developer to access the low level stuff. Three.js is very powerful but it lacks good development tools. I the first early days of this thesis I have created a simple Three.js prototype but I have found it very slow for iterative game design which I was going for.

4.1.2 Unreal Engine

Unreal engine is very similar to Unity and at the time of choosing did not have a free commercial license with full WebGL Support. During this years Games Developer Conference they have announced a Unreal Engine to be free but this was too late after I have started to implement the game.

4.1.3 Unity3d

Unity3D is a great multiplatform game engine, thus enables reaching out to more players. Since they have full WebGl support without the need for embedded web player and it has a wide range of great commercial assets, but it lacks great free assets. Unity does not have any native text built-in in-game editor for syntax highlight.

4.1.4 Decision Process

At first, Three.js seemed as an ideal tool to deploy onto the web platform. Focusing only on one web implementation meant multiplatform interface and game with the benefits of modern web technologies. But with Unity3D coming free for indie developers[1] and the new possibility of games deployment directly to web using WebGL[2] without the need of web player made critical change.

4.2 Visual Programming Language Implementation

To start with I have implemented the class hierarchy as it is depicted in figure 3.1. Using this composite structure I have managed to create a text based interpret. The next step was creating class `ProgrammingBlock` That would pair up with the behavior of ICommand in sense of Composition. From this class all other visual programming block classes inherit. Every visual block has its link to an ICommand it is representing, and when the time of compilation comes it is able to fill-up the ICommand structure associated with it, with blocks that are nested within it.

To allow complex blocks like if blocks I have used spliced sprites and I have created several textures that are being attached to each programming block. To nest the objects I have severely used Unity components such as `Horizontal Layout`, `Vertical Layout`, `Content Size filter` and `Canvas Group`.

4.3 Game Implementation

In this section I will describe important parts of the game from the perspective of both UI and UX design and game design itself.

4.3.1 Game Interface Implementation

The game itself also contains user interface. In this chapter I will go through the parts of the interface that will be important for the player to get his hands onto the game. For implementing the UI within the game there have been also several possibilities. For example, very popular Unity asset called NGUI could be used. I have decided to go with the Unity default UI system for implementing the user interface. This will give me strong future compatibility due to the UI system being introduced fairly recently and an advantage of more tutorials available.

[1] Video game developers who are not owned by any publisher
[2] OpenGL based JavaScript API for rendering 3D graphics in browser.

The Initial Game Menu

The initial game menu composes of three different elements. In the figure 4.5 is depicted the main game menu. This menu contains the option to change the game profile. This will be explained further in the text. The option to engage in multiplayer combat and the link to high scores. To check ranking among other players.

Figure 4.1: Default unity loading screen on Android

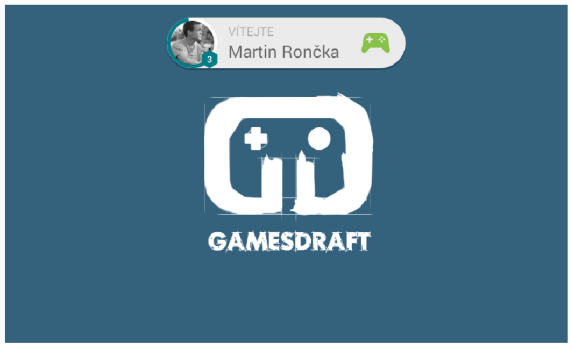


Figure 4.2: Connecting to the Google Play Games service

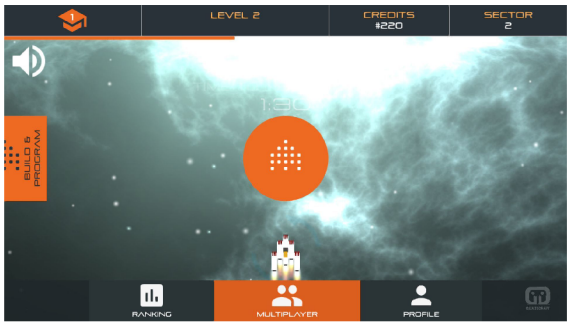


Figure 4.3: Initial game menu with single-player play action and program & build

Figure 4.4: Multiplayer game menu offers competition or cooperation play

Player HUD

This panel shows the player's stats, score and the sector of a game where he ended. This panel is a direct link to developable avatar. The player can see experience progress on the slider under the HUD (orange). This creates a feeling of getting back to where I left off and immediately engages the player.

Figure 4.5: The HUD displays skill points, current level, amount of credits, current sector, and amount of experience

Pause Controls

The game can be paused using the pause button. This button is also triggered and changes state when clicked on the skill button directly from the gameplay as depicted in figure 4.10.

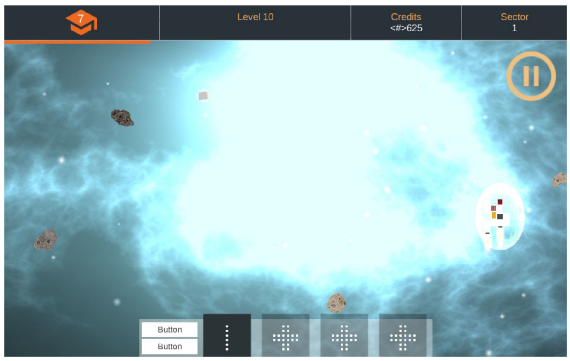


Figure 4.6: Game unpaused, the pause button is hollow

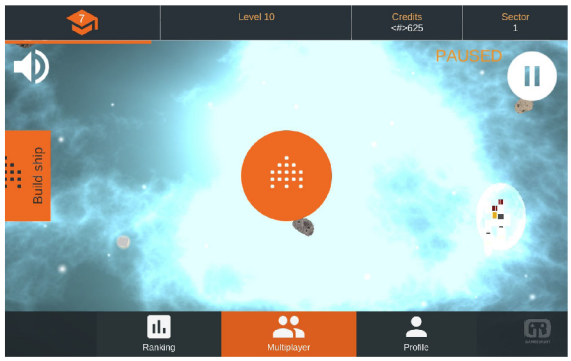


Figure 4.7: When paused, the pause button is filled

Action Buttons

Action buttons are programmable. Once Action button is activated, the code stored within will be compiled into interpretable code which is subsequently executed.

If there is any part of the code that can't be executed from any reason, it get's either skipped or the action stops based on the situation.

Figure 4.8: During singleplayer game, aciton buttons are at the bottom

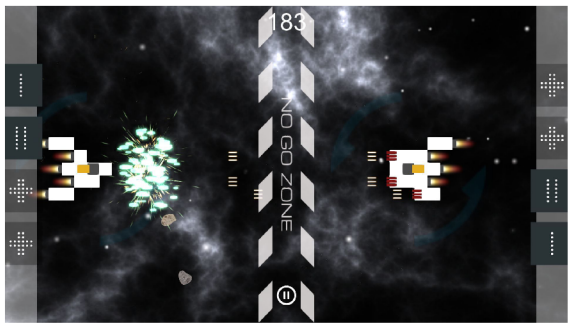


Figure 4.9: In multiplayergame ech player has his action buttons at his side

4.3.2 Multiplatform Input

I have created simple class for handling the input across multiple platforms. Uniting several platform specific actions together as one under the `InputAction` class. This will allow me to change and modify actions at will and give me the opportunity to reuse the class later.

4.3.3 Storing the Player's Data

First we need to store the player's data to persistent memory. Following data will have to be stored to keep the game synchronized over multiple platforms:

- Ship blocks
- Player level
- Amount of experience (points)
- Amount of credits
- Sector where the player currently is
- Player skills (including free skill points)
- Number of programmable slots
- Code for each of the action slots

For this we can use C# serializable notation. We mark our desired classes as `Serializable` and if they meet certain specifications, they can be easily serialized and parsed back. One of the conditions is for example that the class cannot inherit from `MonoBehaviour` which is the default Unity class for components. This means that the classes for storing game data and classes implementing the game logic will have to be independent.

Skills

This menu allows the player to spend the skill points gained by leveling up. Once the skill is selected, the skill name is replaced with a skill icon and a description is shown. After adding, the skill will be no longer addable and will change it's color from dim gray to variety of blue, thus distinguishing from non added skills. If there skills in the tree that are dependent upon added skill, they are unlocked (become selectable) and change their color from transparent gray to dim gray.

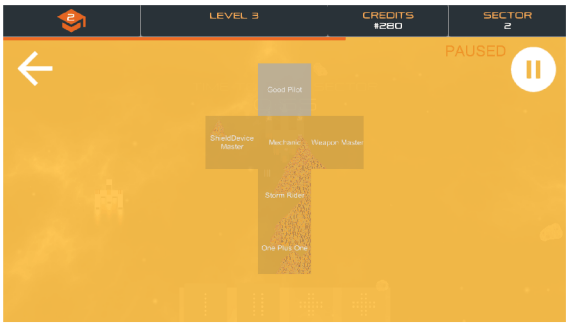


Figure 4.10: Basic skill tree with one addable skill

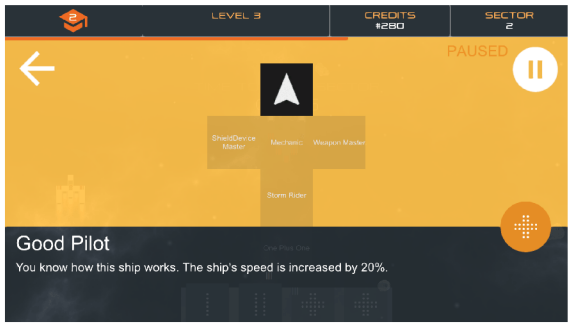


Figure 4.11: Selecting skill shows a description interface

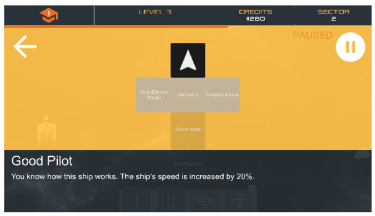


Figure 4.12: After adding a skill, others are unlocked

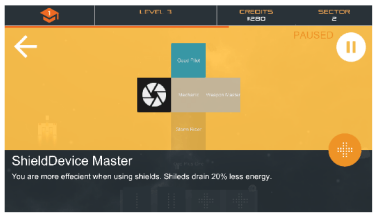


Figure 4.13: The added skill is distinguished by its color

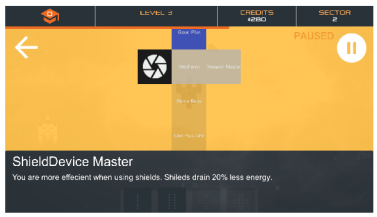


Figure 4.14: The skill tree is scrollable

4.3.4 Reward System

To implement reward system as discussed in section 2.4.1 (e.g. achievements and leaderboard). I have decided to use Google play service which enables to define specific achievement and leaderboard data which are accessible through unified interface on many ported platforms. The implementation itself means installing the Google play games service asset and using the API to:

- Log in
- View achievements,
- Unlock achievements at specific points in the game
- View High score leaderboards
- Commit High score at the end of the game

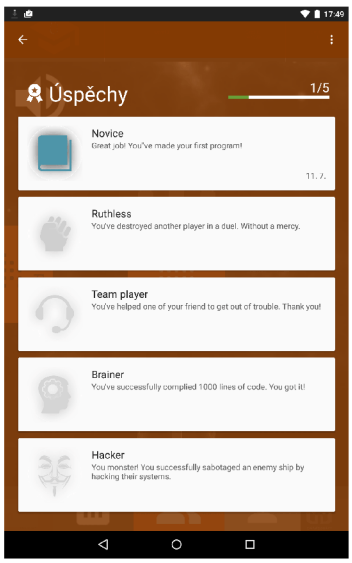


Figure 4.15: Achievements displayed on Nexus 7 in Portrait mode

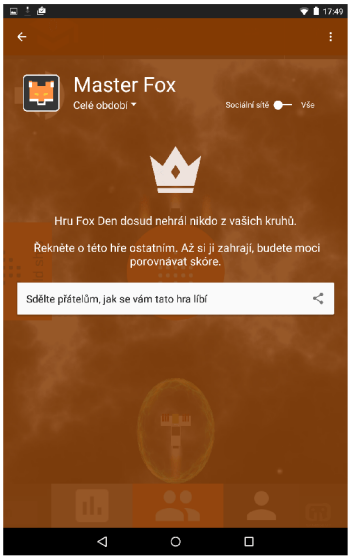


Figure 4.16: Empty leaderboard encourages to share the game

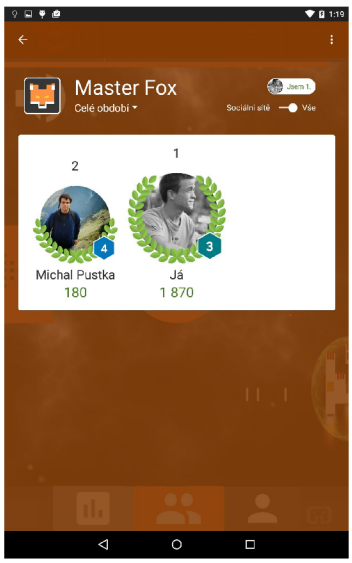


Figure 4.17: Overall leaderboard; highscores within social networks

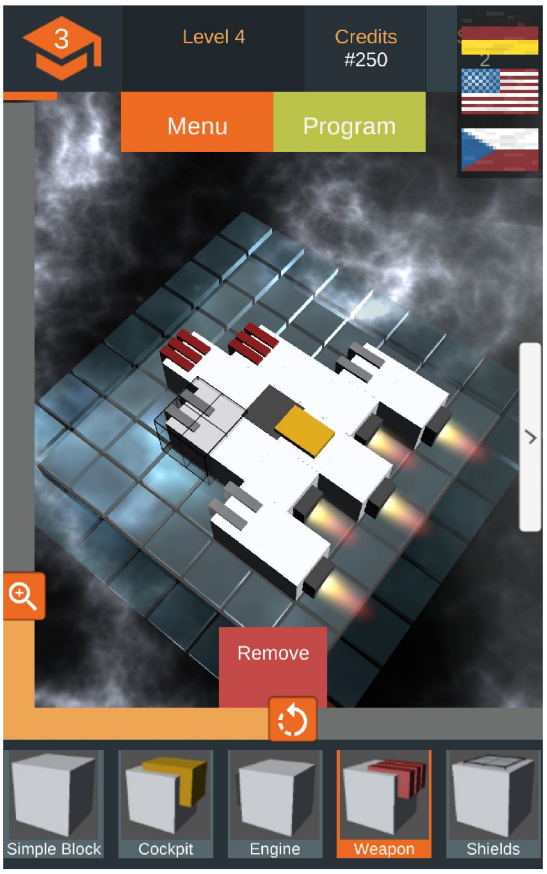


Figure 4.18: Control panel before is by default set to English

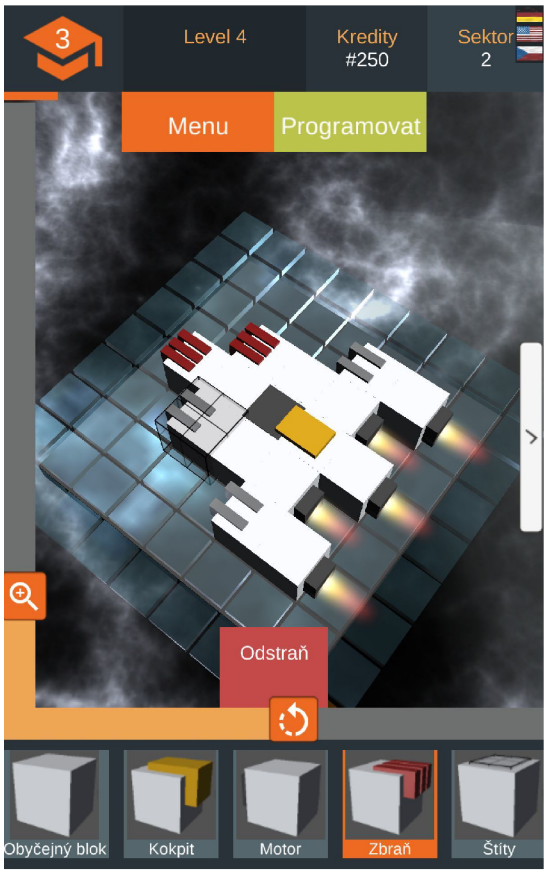


Figure 4.19: Control panel is translated after clicking the Czech flag

4.3.5 Localization

In order to maximize the reach of the game I have decided to implement localization. For this reason I have searched out a very simple localization system for Unity *SmartLocalizaiton*.

This asset enables to create dictionary in any language and enables to change the language dynamically at runtime. After clicking on the flag button at the top right corner, all localized strings are replaced with their variation in given language. As you can see in the figure 4.18 and figure 4.19

4.3.6 Responsive design

To support as many platforms as possible, responsive design is necessary. For this reason Unity offers several tools. Using these tools I have designed a UI that is suitable for recommended types of displays of various sizes and aspect ratios.

First of the tools is anchors. Anchors are part of the UI Rect Transform (main UI positioning component) and is available for each UI element. Anchors are parent of the UI element and allow the UI element to stretch or maintain its proportional position relative to its parent.

Second tool is Canvas scaler which allows to set scaling mode based on the needs of UI elements nested in the specific canvas. Canvas scaler are used independently for each

canvas and thus allow clean separation of given UI elements.

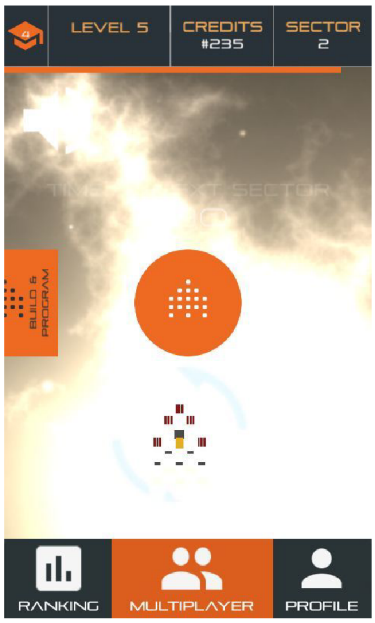


Figure 4.20: Phone vertical

Figure 4.21: Phone vertical

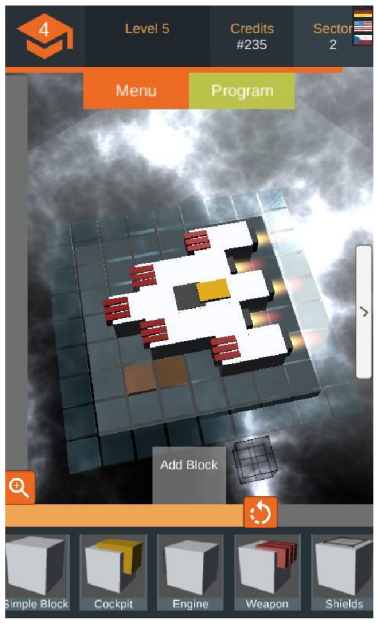


Figure 4.22: Phone vertical

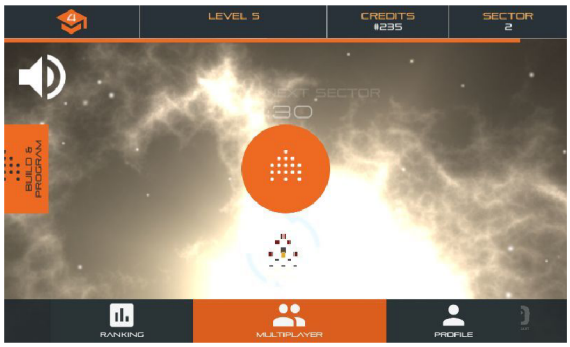


Figure 4.23: Phone; Landscape; Game Menu; Play button; Sound control

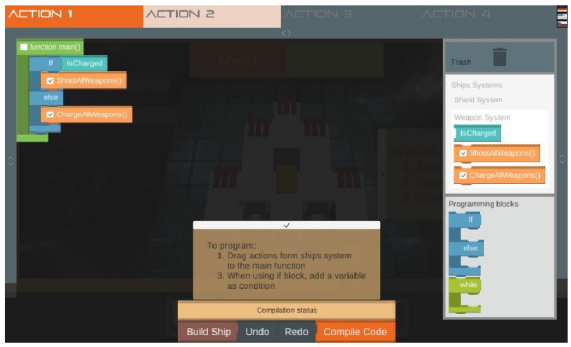


Figure 4.24: Phone; Landscape; Programming interface

Figure 4.25: Tablet; Landscape; In-game; Action buttons; Pause Button

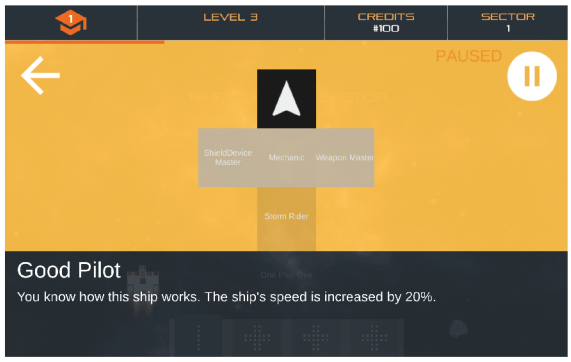


Figure 4.26: Tablet; Landscape; Skills menu with description

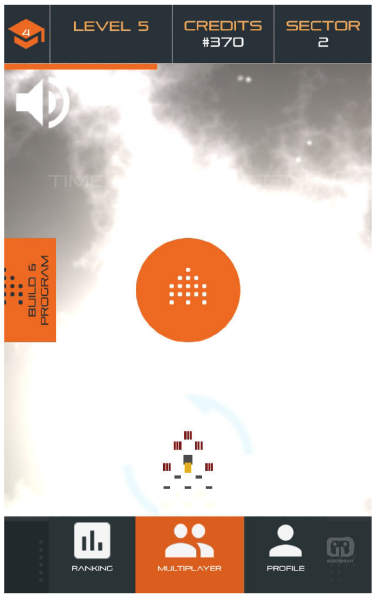


Figure 4.27: Tablet vertical

Figure 4.28: Tablet vertical

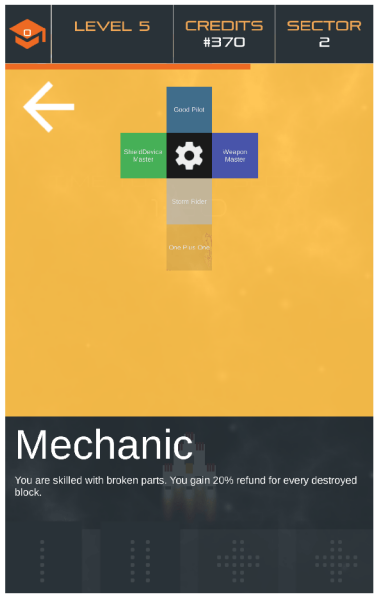


Figure 4.29: Tablet portrait

Chapter 5

Evaluation

In this part I will describe the results of test specification and player feedback on game features and user interface. The evaluation is structured into two main sections. Firstly, it is the evaluation of game and user interface by humans. And secondly, it is the evaluation from technical perspective, the *frame rates per second (FPS)* on specified platforms, the distribution package and the file sizes of binaries for each platform.

5.1 Human evaluation

I have given the game to 5 people with several specified tasks. All the additional testing data, forms and photos can are stored on the dvd (more at the appendix A. Following tasks have been given to the players:

Task 1 *Get to the second sector.*

Task 2 *Build weapons and program the action button to:*

- *Shoot if the weapons are charged*
- *Charge if the weapons are not charged*

Task 3 *Add following behavior to the program:*

- *Deactivate shield before shooting*
- *Activate shield before charging weapons*

Task 4 *Optional tasks:*

- *When leveling up, choose skill good pilot and add it*
- *Build nice ship and send the picture*

The following table shows how well have they done:

Task	Success rate
1	100%
2	100%
3	80%
Optional 1	60%
Optional 2	60%

5.1.1 Reactions

In this subsection there are listed answers to questionnaire which was distributed with the game. For full and original responses look at the appendix A for content of the DVD file.

Did the Game Run Fluently

- Yes, it did. No glitches.
- The Game run fluently.
- Yes.
- Yes.
- Yes.

did anything special happen during the gameplay

- I figured out how to be invincible. I kept pressing the 'space' button, which turned on the shield and my enemies couldn't get to me. I observed their inevitable destruction by meteors.
- Nope.
- I do not think so.
- Nothing strange happened.
- I died 500 times.

Did the programming blocks work as expected?

- Once I figured out how they work, they did.
- yes but the green ones were a bit of a breakneck.
- Yes, they functioned pretty well but once I could not immediately move it to the right place, so I put it into trash and tried again.
- Sort of...
- Blocks worked well.

Did you use undo and redo buttons, optionally trash?

- Only the 'trash' function. It didn't work well at first, but then it was okay.
- I did not use any of these.
- I used the action back and also the action trash.
- Yes.
- Trash yes, undo and redo no.

Did you like the visual side of the programming interface?

- Yes, I think that this is creative way how to learn programming. I have already had the orders in action so I could not add it. I have just compiled. I like the possibility to built the ship in various ways. The multiplayer is much fun :D.
- The visual side is nice, and possibility of turning the ship is good.
- Yes, they were cute.
- I did, but the UI needs more improvements, pictographs are not intuitive, some of the namings were misleading.
- Jop

5.2 Programming Catalog

Here I would like to summarize what programming concepts this game and the VPL created during this thesis currently offers to practice:

- Object orientation
  - Object abstraction
  - Member variables
  - Member methods
- Sequential programming
  - Invoking member methods
  - Executing Conditional statements

These are not many concepts but the big advantage is that the VPL is written i general way and so its is possible to be deployed as an unity asset and extended furthermore. Yet the game could use in future some more programming constructions, sadly I have not been able to find a way to incorporate cycles.

5.3 Technical Evaluation

5.3.1 Distribution

Unity allows quick deployment for multiple platforms, I have successfully compiled binaries for 5 different platforms for both 32 and 64 bit versions of the application. There has been only one issue and that was during the compilation for Web using the new direct WebGL build. This has been acknowledged by Unity development team as a Bug of current `Unity version 5.1.0f3`.

Platform	Binaries files size
Windows	`18,4 MB`
Linux	`61,8 MB`
Mac OS	`69,4 MB`
Web	`6,70 MB` - requires web player installed
Android	`30,1 MB`

Unity also allows to split up the android .apk into extensions an .apk and extension in form of .obb file. But the game assets are negligible in comparison of today's hardware.

5.3.2 Frame Rates

I have written simple script to watch over the FPS within the game. In the following table i will list the devices the game has been tested on and the average FPS the game ran at.

Platform	CPU	GPU	FPS
Windows 10	i5 3570k	GeForce GTX 660	Stable 300
Windows 10	I3-3217U	HD Graphics 4000	Stable 80
Laptop Windows 8	i5-3210M	GeForce 630m	40 Unstable due deprecated drivers
Laptop Windows 7	I7-4500u	HD Graphics 4400	184
Windows 7 Enterprise SP 1	Intel R Core TM Duo P8700	Mobile Intel(R) 4 Series Express Chipset Family	70
LG V500 Android 4.4.4	ARM Cortex A7	GHz Adreno 320	60
LG F60 Adroid 4.4.4	Qualcomm Snapdragon 410	Adreno 306	60
Nexus 7 Android 5.1	Qualcomm Snapdragon S4 Pro	Adreno 320	60
Nexus 4 Android 5.1	Qualcomm Snapdragon S4 Pro	Adreno 320	60

This table clearly shows that the game is very well optimized and is capable of running at vast amount of devices. Considering the Android platform, Unity allows compatibility up to Android 2.3.1 Gingerbread. The android developer console shows 9252 o of supported devices which is 92.4% of device types currently market devices.

Chapter 6

Conclusion

Playing games in today's world is very common. If we managed to move from playing regular games to playing programming education games for just a small portion, I believe that we would end up with kids having much more positive attitude towards programming.

I have conducted a research regarding existing serious programming education games and from the analysis I took that the market is huge with a large amount of similar games. Majority of these games are still focused more on the basic sequential programming. There is also large portion of education tools using visual programming languages to that focus on the object oriented programming.

I have studied large amount of User interfaces with focus on Android platform and I believe that I have managed to create a nice UI template for future use, considering future improvements. I have managed to create simple but visually attractive visual scripting language that can be used in multiple projects, and thanks to Unity, is available on every supported platform. The interface was tested out on three platforms and behaves correctly at all tested cases. As one of the main lags of this thesis I consider the small amount of experiments conducted due to the late implementation and higher focus to the uniqueness and range of education elements embedded.

I have designed a game and associated interface with regarding to modern mobile games. I have implemented responsive user interface that has been tested out on multiple platforms and incorporated this user interface into the game. I have designed and implemented visual programming language that represents the games educational value. The UI is simple but sophisticated and allows for future work within the field of visual programming. And finally I have evaluated the game using human feedback and as the result I find the interface very interesting as a concept, but with more work needed to be done.

There are several opportunities for future work. Both the game and the programming interface can be extended severely. For example the game could use online multiplayer to engage players in cooperation mode and to extend the competition mode to motivate them even without people around them. The amount of ship blocks that are offered is also very limited and could use an iteration. Finally the visual programming interface could be extended further into an independent tool much like Blockly and distributed as a Unity asset.

Bibliography

[1] Tiffany Barnes, Eve Powell, Amanda Chaffin, and Heather Lipford. Game2learn: Improving the motivation of cs1 students. In *Proceedings of the 3rd International Conference on Game Development in Computer Science Education*, GDCSE '08, pages 1–5, New York, NY, USA, 2008. ACM.

[2] Iván Cantador and José M. Conde. Effects of Competition in Education: A Case Study in an E-Learning Environment. In *Proceedings of IADIS International Conference e-Learning 2010 (E-Learning 2010)*, Friburg, Germany, July 2010.

[3] Mark Durando. Computing our future. European Schoolnet, October 2014. Available at http://www.eun.org/c/document_library/get_file?uuid=521cb928-6ec4-4a86-b522-9d8fd5cf60ce&groupId=43887.

[4] Neal Hallford and Jana Hallford. *Swords and Circuitry: A Designer's Guide to Computer Role-Playing Games*. Premier Press, Incorporated, 2001.

[5] Wang Hao and Sun Chuen-Tsai. Game reward systems: Gaming experiences and social meanings. In *DiGRA &#3911 - Proceedings of the 2011 DiGRA International Conference: Think Design Play*. DiGRA/Utrecht School of the Arts, January 2011.

[6] Dan Ingalls, Ted Kaehler, John Maloney, Scott Wallace, and Alan Kay. Back to the future: The story of squeak, a practical smalltalk written in itself. *SIGPLAN Not.*, 32(10):318–326, October 1997.

[7] Alan Kay. Squeak etoys, children & learning. 2004. Availible at http://www.squeakland.org/content/articles/attach/etoys_n_learning.pdf.

[8] S. Vygotsky Lev. Play and its role in the mental development of the child. *Soviet Psychology*, 5(3):6–18, 1967.

[9] Richard E. Pattis. *Karel The Robot: A Gentle Introduction to the Art of Programming*. John Wiley & Sons, 1981. ISBN 0-471-59725-2.

[10] Marc Prensky. *Digital Game-Based Learning*. McGraw-Hill Pub. Co., 2004.

[11] Viera K. Proulx. Computer science in elementary and secondary schools. In *Proceedings of the IFIP TC3/WG3.1/WG3.5 Open Conference on Informatics and Changes in Learning*, pages 95–101, Amsterdam, The Netherlands, The Netherlands, 1993. North-Holland Publishing Co.

[12] Filip ŠELONG. Game-based learning: Příklady dobré praxe a návrh výukové počítačové hry [online], 2012 [cit. 2015-07-28].

Appendix A

DVD Content

DVD content is as follows:

- `doc/` - Technical report, pdf version and latex source files with images
- `src/` - Source files of the unity project including all assets and images.
- `bin/` - Binary files
  - `Android/` - Android .apk file
  - `Windows/` - .exe file with data
  - `Web/` - .html file and Unity web player data
- `testing/` - All files regarding testing
  - `Questionary/` Testing questionnaires with responses
  - `Notes/` Testing notes on further enhancement
  - `Images/` Images and screenshots from testing

Appendix B

Installation

The following binaries present on the DVD have been successfully tested:

- Windows .exe x86 and x86-x64
- Android full .apk
- Web Online.html

If you are interested in playing or testing the game, it is available online either through the open alpha/beta testing group or directly on the Google Play store. To join the alpha/beta testing go to the following link:

- https://play.google.com/apps/testing/com.GamesDraft.FoxDen

Confirm by clicking on „Become a tester“ button. You can leave the testing group at will at any time.

Appendix C

Posters

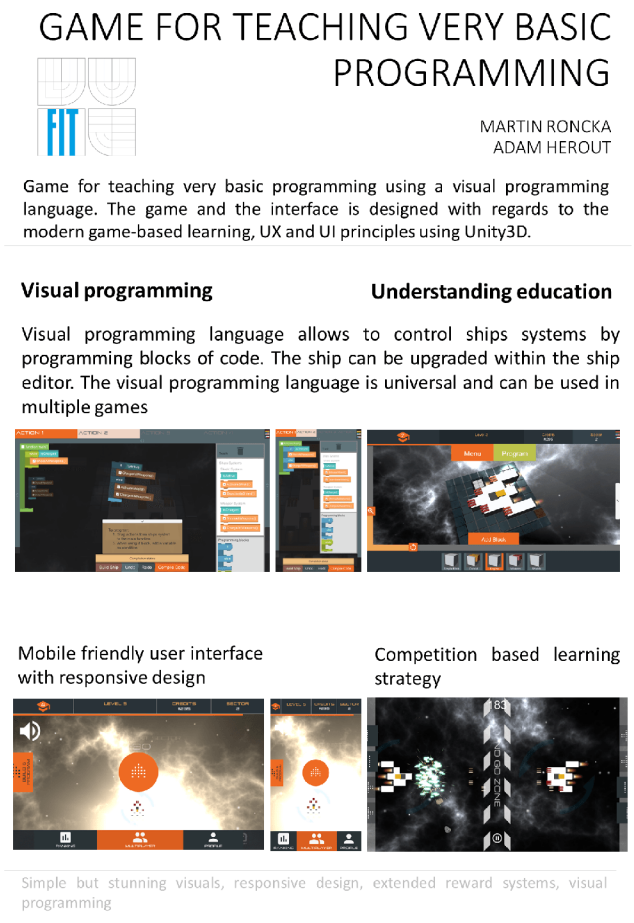


Figure C.1: Presentation poster