Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

**Oriol Tobar**

# GAME DEVELOPMENT - FROM THE VISION TO THE FINAL PRODUCT

Examiners : Doctor Jussi Kassurinen
Doctor Ari Happonen

Supervisors: Doctor Jussi Kassurinen

# ABSTRACT

Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Oriol Tobar

**Game development - from the vision to the final product**

Bachelor's Thesis

32 pages, 11 figures, 7 tables,

Examiners and supervisors: Doctor Jussi Kasurinen
Doctor Ari Happonen

Keywords: academic thesis, game development, programming, art, game design document

The main objective of this bachelor's thesis work is to find out if a single computer science without background in the game industry can create a full playable game or at least a proof-of-concept without any external help. The results are successful because the student created it and also the general aspects of game development are shown and discussed. The main constraints are the lack of knowledge in every field of game development and also the time, which it is mandatory when creating a good game. This work is pretended to be helpful to future students in the same situation of the researcher of this project that want to work as game developers. This study has been done as part of the Erasmus program in the Lappeenranta University of Technology.

**ACKNOWLEDGEMENTS**

I would like to thank my family and all my friends that helped me in this long journey.

*Players are artists who create their own reality within the game.*
-Shigeru Miyamoto

*The only way to discover the limits of the possible is to go beyond them into the impossible.*
-Arthur C. Clarke

# TABLE OF CONTENTS


1. INTRODUCTION 7
   1.1. Background 7
   1.2. Goals and delimitations 7
   1.3. Research questions 8
   1.4. Structure 8
2. LITERATURE REVIEW AND BACKGROUND 9
   2.1. Skills for game development 9
   2.2. Risks and problems 10
3. METHODOLOGY 10
   3.1. Development process 10
   3.2. Project Plan 11
      3.2.1. Requirements 12
      3.2.2. Specifications 12
      3.2.3. Project Scope 13
      3.2.4. Project structure 13
      3.2.5. Work packages and schedule 14
      3.2.6. Risk analysis 15
   3.3. Game design document 16
      3.3.1. Introduction 16
      3.3.2. Background 16
      3.3.3. Game Concept 16
      3.3.4. Story 17
      3.3.5. Characters 18
      3.3.6. Gameplay 18
      3.3.7. User Interface 19
      3.3.8. Weapons and armours 19
      3.3.9. Genre 20
      3.3.10. Levels 20
      3.3.11. Extras 21
   3.4. Implementation phase 21
      3.4.1. Graphics and animations 21
      3.4.2. Programming 23
   3.5. Testing 26
4. RESULTS AND CONCLUSIONS 27

4.1. Project management 27
4.2. Game implementation 28
4.3. Research questions 28
5. FUTURE WORK 30
5.1. Publishing the game 31
6. REFERENCES 32

## LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| AI | Artificial intelligence |
| CD | Collision Detection |
| GDD | Game design document |
| GUI | Graphic user interface |
| HUD | Heads up display |
| RPG | Role playing game |
| PNG | Portable network graphic |

# 1. INTRODUCTION

Video game development became very popular in the last years and every single day there are more and more students that are interested in this industry and how to get in.

With all the possibilities offered by Internet actually is more accessible to self-learn how to create them but there is still a big gap between the knowledge acquired during the bachelor or master and the real skills needed in an AAA studio or even a little studio.

This thesis will go throughout the whole real process of creating a game by a single student without background in the game industry and will be a kind of work diary explaining all the difficulties found and the skills needed to do the game.

## 1.1. Background

This bachelor's thesis is the final project of an Audiovisual Engineer of the Polytechnic University of Catalonia in an Erasmus program in the Lappeenranta University of Technology under the supervision of the professors of the School of Business and Management. This student wants to do the Master in video games so by doing this project he will achieve some experience in the game industry and at the same time learn how the typical game development process works.

## 1.2. Goals and delimitations

The main objective of this project is to study what skills and resources are needed in the game development process for a single student with a medium-high programming background and high image and audio processing. The case of study will be the development of a small 2D playable game, by doing this the student will find out which are the hardest parts of the development, which are the easier or which are already familiar for him.

The main delimitation will be the time, this project has duration of 5 months and within this time he has to be able to go throughout the different development phases and get over the difficulties. Other delimitations are the lack of knowledge in sprite drawing or game designing but these are easier to lead using Internet or reading books.

## 1.3. Research questions

There is a main research question that the student will try to find out the solution:

> ***Can an Audiovisual Systems Engineer cover all the areas involved in a game development project? From the idea to the final product***

This is a huge question to answer so it will be divided in a set of sub-questions:

1. *Can I use my previous knowledge in signal processing to create efficient graphics and audio? How my tools will match in the game development world*
2. *How games are designed, developed and tested?*
3. *Is it hard to a student without previous experience to get into the world of games? Will I be capable to learn how to use the different tools and software involved in a game development process?*
4. *If the game is finally published, can I monetise it?*

In order to answer all these questions the student will create from the scratch a playable 2D platformer game for Android.

## 1.4. Structure

The structure of the thesis will be divided in 6 different parts:

- Section 1 contains the introduction with the main objectives of the project, research questions and some background about the student.
- Section 2 is about the literature review and the current state of art of game development involving inexperienced students and the problems that can appear during the process.
- Section 3 is the most important; it contains the methodology of the project; which includes the project plan, the game design document and the implementation phase.
- Section 4 and 5 involves the final results acquired in the project and conclusions with the possible future work to do.

Finally, the section 6 includes the references and the bibliography used in the project.

# 2. LITERATURE REVIEW AND BACKGROUND

The first computer games appeared in the sixties and seventies of the 20th centuries, who can imagine that these interactive classic games like chess or pong will become one of the biggest industries in the world? This industry generates more money than the film industry.
With the growth of the mobile devices and more powerful computers and consoles the consumers are interested to pay for subscriptions services or gaming services, although this is not quite popular in mobile devices is becoming more popular for the companies to include in-app purchases to their product while the game is free for download. Also, many consumers want their consoles for more than just gaming, for this reason the companies are maximising other aspects like video or audio on demand [1]. For all this reasons is possible to say that video games have become part of the culture of the people hence the benefits from them are only growing more and more every year.

Game development can be divided in three phases: preproduction, production and post-production. The first phase consists to evaluate all the ideas available to until one stands out and then the game proposal is created. After that different prototypes are created and played internally to get feedback in order to help the designers and managers to understand which one is the best and fits well in a commercial perspective. The production phase starts when all the team knows what the want to build and how they plan to create it, this means that they can start to create real assets that will be used in the game. Finally, post-production is the final phase of the game development cycle and this involves the marketing strategies and final distribution of the product. [2]

Beginners in game development such as computer science students are capable to develop a game? [6] The current curricula of computer science degree or other technical degrees with programming background are not enough to give the skills necessaries for the game development. Students tend to focus on technical objectives not on game experience and they are not ready to do changes if a problem appears during the development.
Approximately half of the topics needed to develop a game are not covered in the universities computer science degrees [6], for example the most important topic which is 3D modelling is not covered or audio work.

## 2.1. Skills for game development

There are a lot of skills needed for game development, that is the reason of a studio to have thousands of workers. These skills can be: game designing, programming tasks, graphics creation, sound engineering, testing or business. In an Indie studio formed by little teams like 5-20 workers some of them can manage to do various tasks at the same time. When a student wants to get into the game industry has to think about which area he will fit well, normally a computer science student is supposed to be good in the programming part or a digital art and animation student should fit well in the art department.

## 2.2. Risks and problems

J. Blow [3] said that programming video games became very hard mainly because the new consoles are more complex and the project size of a game. Blow said that for example, many of the tools existing around 2004 don't help to game development because are not aimed to game development. Third-party components [3,5] usually offer a good solution because it can reduce the workload heavily, usually this third-party components could be the game engine for example. Game engine saves to the programmers about half millions of line codes [5].
Nowadays is it possible to say that the game developer companies are pleased with the current tools that are available [5], one important fact is that with this tools the developers can make fast changes in the design furthermore these tools are important in the product design and prototyping.

# 3. METHODOLOGY

This section contains the development process of a playable video game in 2D based on platforms, this is a kind of working diary collecting all the problems and tasks that the student had to face during the project.
The game is a proof-of-concept due to the lack of time and knowledge, time is something that we can't modify but knowledge can be solved reading books, papers or Internet. Thus, the research method used in this work is by reading books and then watch videos of video game art in order to achieve some background in every field and then start with the development of the game.

## 3.1. Development process

The development process is something that every studio has to follow if they want to create a good game. This process is iterative, this means that some features could sound good when written in papers but then when created are not working properly or they are not as good as expected.
Being iterative is good, this means that the developers are always trying to do their best fixing or updating the initial plans to create a better game.
The game development is divided in 5 different phases:

- **Game design:** In this part the game designer *creates* the game by writing a document called Game Design Document (GDD) which is the bible that every member of the team will read and follow to create what the game designer wants exactly. In the end of this phase the team should come up with some drawings of the final game or even a playable prototype.

- **Implementation:** This phase is also called *production* and is the main stage of the development. Assets and source code for the game are produced. In this part the team is fully staffed with programmers that start with source code, graphics designers creating the first assets and sound engineers creating the first sounds. This phase usually ends with an *alpha* version of the game. This *alpha* is a feature

complete game, this means that it contains all the major features and only some changes or improvements will be added, unimplemented features may be dropped.

- **Beta phase:** This version is feature and asset complete with only bug fixing from the feedback of the users or beta testers. No new code is added and when this phase is reached it means that the game is fully playable with no minor bugs.

- **Post production:** Finally in the post production the game is shipped and is when the business members of the team start with the publicity and the mass media to advertise the game. With the popularity of the online console games there is a *maintenance phase* in which the team waits for feedback from the users and release patches with bug fixing or adding new features.

The next figure shows how is the development process and his iteration in the production phase:

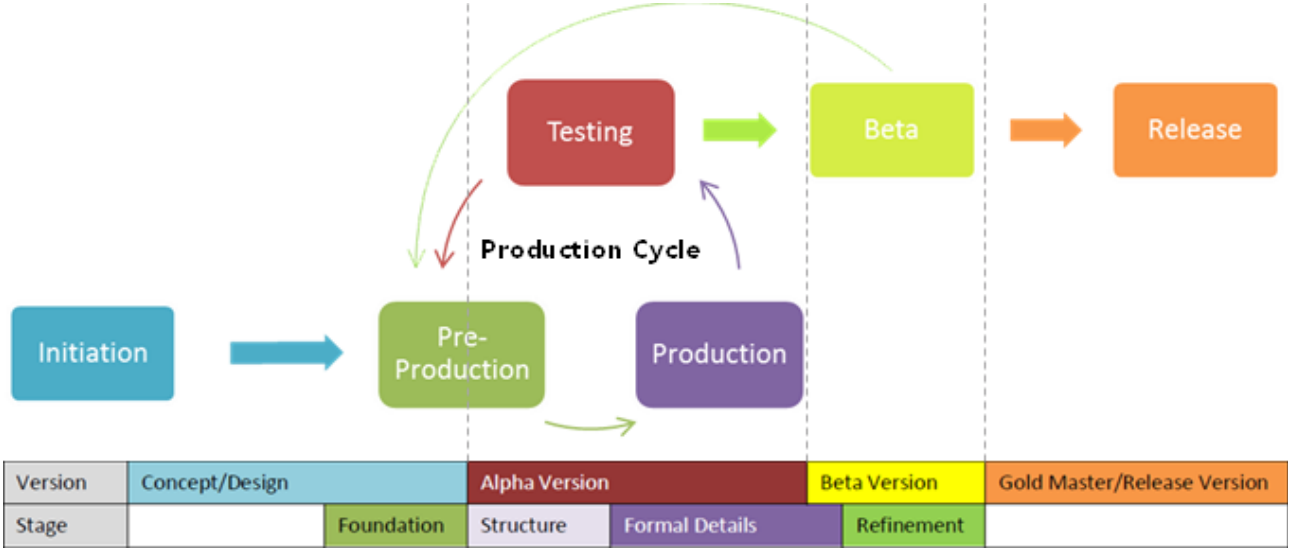



Figure 1. Game development process

In this project with only 1 member all the roles in a studio are represented by the student, at first he will read books about game development to cover the game designing phase, watch tutorials in YouTube or Udemy to learn how to draw the sprites and programming with Unity. After this first iteration the game itself will be created and tested.

## 3.2. Project Plan

The project plan indicates the main objectives of the project as well as the milestones, specifications or responsibilities. It is very important to follow the project plan in order to achieve a successful work.
As mentioned, this plan defines everything involving the game and milestones but it can be modified in the future with a critical review, evaluating if the milestones are completed correctly or there is lack of time.
These are the different parts of the project plan.

### 3.2.1. Requirements

The main project requirements will be to learn how works the process of designing and developing a video game. All the phases involved in the game development cycle like preproduction, production and postproduction.
More requirements expected of the project is to learn how to draw graphics and compose the music for the game, scripting, artificial intelligence, level creation and user interface.

### 3.2.2. Specifications

As this is not a project with physic measurable things, the specifications will be focused on the software needed to develop the game. The reasons for choosing these tools are personal preference and compatibility with other tools mainly also C# has got huge community support and libraries for Unity.

The next table is a summary of all of them:

| Software | Purpose |
|---|---|
| Unity3D | Main tool, this will be the game engine. Cross-platform |
| Adobe Photoshop | 2D pixel art graphics creation |
| MonoDevelop | Scripting in C# |
| Garageband or Famitracker | 8 Bit music and audio effects |

All these tools will run in a MacBook Pro mid 2010 with the next technical specs:

| |
|---|
| 2,4 GHz Intel Core i5 |
| 8 GB 1067 MHz DDR3 |
| NVIDIA GeForce GT 330M 256MB |
| 15,4” panel with 1440x900 resolution |
| Crucial M4 SSD 256GB storage |
| OSX 10.11.1 El Capitan |

### 3.2.3. Project Scope

The project will be based on creating a platformer game with role playing game (RPG) features for portable devices, the in-scope features are:

- Single player
- Android or iOS devices
- 2D pixel art
- Multi level
- Platformer with RPG elements such as NPCs or shop with weapons and armours
- Touch Screen GUI based on pixel art
- Missions and secrets
- Unity3D using C# as programming language

### 3.2.4. Project structure

The next table includes the different phases with the deliverables and the date of presentation:

| Phase | Deliverable | Date |
|---|---|---|
| Game Design | Storyboard, characters and other elements like the GDD | 6/10/2015 |
| Implementation | 2D assets, scripting, artificial intelligence, collision detection | 12/11/2015 |
| Alpha | Boss level and improvements like smoother animations | 18/12/2015 |
| Beta | Touchscreen UI & Menus, testing | 30/12/2015 |
| Post production | Final report | 4/01/2016 |

### 3.2.5. Work packages and schedule

The next tables are the different work packages with the internal tasks and the schedule planned for the development process:

| Project: **Game design document** | WP ref: (WP1) | |
|---|---|---|
| Major constituent: **Document** | Sheet 1 of 5 | |
| **Short description:**<br><br>Document that explains in detail how the game play elements will interact and what the game world and characters look like | Planned start date: **1 Oct**<br>Planned end date: **11 Oct** | |
| **Internal task T1**: Write the storyboard<br><br>**Internal task T2**: Define the characters, enemies, weapons and other elements present in the game<br><br>**Internal task T3**: Level design (3 levels + boss stage), only first draft | | |

| Project: **Implementation** | WP ref: (WP2) | |
|---|---|---|
| Major constituent: **Software** | Sheet 2 of 5 | |
| **Short description:**<br><br>Creation of the basic 2D pixel art graphics for the levels, characters, weapons, items and other elements. First playable prototype with at least 1 level included and basic IA and collision detection interaction | Planned start date: **12 Oct**<br>Planned end date: **29 Nov** | |
| **Internal task T1**: Graphics creation<br><br>**Internal task T2**: Scripting in C#<br><br>**Internal task T3**: Basic IA and collision detection<br><br>**Internal task T4**: Basic animations of the characters | | |

| Project: **Boss level & improvements** | WP ref: (WP3) | |
|---|---|---|
| Major constituent: Software | Sheet 3 of 5 | |
| Short description:<br><br>Creation of the boss level. Enhancement of the first level and sound effects implementation. Creation of some other new sprites like 1 weapon, 1 armor & other enemies (at least 1 more) | Planned start date: **2 Dec**<br>Planned end date: **23 Dec** | |
| **Internal task T1**: Creation of the Boss level<br><br>**Internal task T2**: Sound effects and extra animations<br><br>**Internal task T3**: Creation of extra sprites | | |

Beta phase:

| Project: **Touch Screen User Interface** | WP ref: (WP4) | |
|---|---|---|
| Major constituent: Software | Sheet 4 of 5 | |
| Short description:<br><br>Porting the game to Android devices, which means the implementation of the touch screen controls. Implementation of the *Main Menu*, *Shop* and *Settings*. | Planned start date: **27 Dec**<br>Planned end date: **3 Jan** | |
| **Internal task T1**: Touch screen user interface<br><br>**Internal task T2**: Implementation of the Menus | | |

| Project: **Post production** | WP ref: (WP5) | |
|---|---|---|
| Major constituent: **Documentation** | Sheet 5 of 5 | |
| **Short description**:<br><br>Testing the game with real users to get feedback and bug testing.<br>Final report of the whole project | Planned start date: **4 Jan**<br>Planned end date: **24 Jan** | |
| **Internal task T1**: Testing and feedback<br><br>**Internal task T2**: Write the final report of the project | | |

### 3.2.6. Risk analysis

There are always different type of risks present in a game development process, the main risk for a single student doing a game is the time. With a limited amount of time the student has to be capable to learn and understand all the elements involved as well as the creative part that is usually the hardest because of the lack of knowledge in this field.

These are the different type of risks expected in the development and the priority to solve them:

| Risk | Impact in the development | Possible solutions |
|---|---|---|
| Milestones (time) | Very high | Work hard during all the project and try to follow the deadlines |
| Assets creation | High | Watch tutorials on YouTube and Udemy about the topic |

| | | |
|---|---|---|
| Project plan too ambicious | High | Reduce the levels, items or other aspects planned initially in order to have lower workload |

## 3.3. Game design document

The game design document (GDD) is the *soul* of the game. The designer explains in extreme detail how the game play elements interact and what the game world characters look like. This document is a reference for programmers, art designers, sound engineers and the rest of the team on how to do the game. The GDD is a "living" document that might grows and evolves over time. [10]

Each company has its own approach to structuring the document; in this case the student will try to create my own structure the most coherent and continuous as possible.
The background section is optional in GDD because it explains if the game concept is expanded upon other products, my game is strongly inspired in other titles so its necessary to include the background.

---

### 3.3.1. Introduction

*Sword of crystals* is an action platform game for Android developed in Unity3D with retro but colourful pixel art style. With simple controls the player will adventure through 3 different levels full of monsters collecting gems to build the different crystals or find the hidden chests looking for better items.

Beat Yggdrasil in the final stage with the mighty Sword of Magic made by the crystals and save the world from darkness!

---

### 3.3.2. Background

*Sword of Crystals* is clearly inspired in two existent titles already available in iOS.
These games are called *Goblin Sword* and *Sword of Xolan*, both are the same genre that *Sword of Crystals*.

This game is inspired in these tittles because pixel art is a retro view for games but very popular nowadays and this kind of platform games are quite addictive to these players who only want short gameplays while they are waiting the train for example.

---

### 3.3.3. Game Concept

As it is mentioned, *Sword of Crystals* is based on *Sword of Xolan* and *Goblin Sword*, which motivated the student to create this proof-of-concept and try to find out if an Audiovisual Engineer student is able to create *from a vision* a similar game.
The only features added is the history and certain weapons or items that not appears in the other 2 tittles.

The game concept (or research question) is to find out if a single student is able to develop the entire game without help in the scheduled time.

### 3.3.4. Story

According to an old legend there are three mystic crystals that keep the equilibrium of the world, each crystal holds one power inside. The green crystal holds the power of nature; the red holds the power of fire and the black the power of death. One day one mighty hero will come and collect them to build the Sword of Magic and free the world of darkness.

Now the world is plunged into darkness because of Yggdrasil. He is a dark magic knight and the king of Baronia. He wants to rule the world by spreading death and dark powers throughout it.
Elkyy, a young boy from Karhillania seems to be the mighty hero of the legend, he is brave enough to try to collect the three crystals to stand against darkness and bring peace and serenity that once was.

Elkyy will have to fight and beat thousand of enemies from the hordes of Yggdrasil in three different levels (nature, fire and death level) to collect one legendary crystal in each one.
These crystals will not be easy to find, each one is split in 3 little gems hidden in the level, Elkyy has to be smart enough to find the gems and in the end of each level the big one will appears and be collected by him. Also there are hidden chests with secrets and powerful weapons and armors hidden in the map!

Once it is done he will be able to build the Sword of Magic and have the final battle against the dark magic knight. The humanity is running out of time in front of the dark army and Elkyy must be fast and brave to beat them all before Yggdrasil destroys everything.

The next image shows the process of the story:

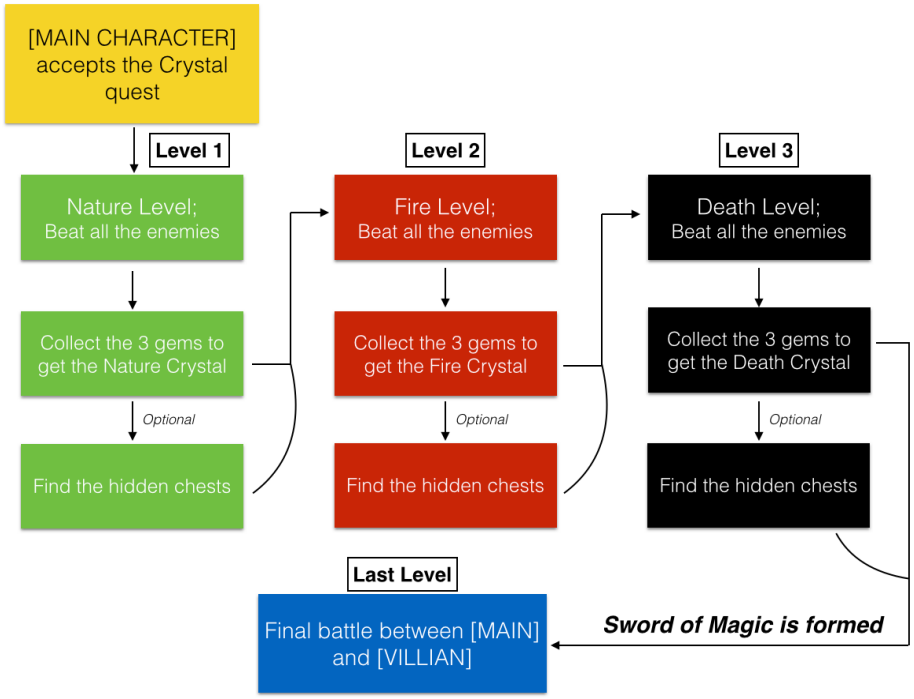



Figure 2. Flow diagram of the story

### 3.3.5. Characters

In this game there will be only two important characters, the main character and the villain. All the other will be creatures (enemies) and a friendly NPC (shop).

- Elkyy: (Main Character) He is a young boy from a small village called Karhillania. He will fight all the enemies no matter what the cost is. At the beginning of the game he will accept the quest of retrieving the mighty crystals and finally he will become the hero of the legend and have the final battle against Yggdrasil.

- Yggdrasil: (Villian) King of Baronia and the last dark knight of the planet. He also possesses a huge dark power inside. The only goal of this cruel king is to devastate the world using his minions and then become the ruler of all the people. The only way to beat Yggdrasil is with the legendary Sword of Magic.

### 3.3.6. Gameplay

The gameplay is Action-RPG mixed with platform style, the player controls the main character; which in the beginning will have 3 hearts (or lifes) that will reduce when entering in contact with an enemy or jumping in the spikes or fire for example.
You have to beat the enemies and collect the 3 gems (or crystal shards) hidden in the level in order to pass to the next one. Only when the 3 are collected the next level is unlocked, otherwise it remains locked in the level selection screen.
Killing enemies will give you coins that you can spend in the shop and buy new items, weapons and armours; enemies can also drop hearts to recover life.
There will be two hidden chests in each level that also will have special items or coins; these chests are not mandatory to collect in order to pass to the next level. (Optional)

The game is composed of 3 levels plus the last stage with Yggdrasil (villain), each level will be inspired in one topic:

- Nature (green)
- Fire (red)
- Death (black)

This means that for each level the enemies and other elements will have the dominant color specified before.
When the player collects the 3 crystals Sword of Magic appears and you can equip to the hero and then the last stage will be only a 1 vs 1 battle between the player and Yggdrasil as the final boss.

There is magic available as well, the player can shoot 3 times the magic power and then he will have to find mana potions dropped by enemies or hidden in the map to restore again all the shoots.
When it comes to weapons and armours, there will be only 4 armours to buy in the shop and they only change the appearance of the hero. The weapons can give extra attack power, speed or range and there will be 4 different types plus the Sword of Magic.

### 3.3.7. User Interface

The player begins in the main menu screen; which has *Options* and *Start*. *Options* will have several buttons to be specified like *Mute*, *Credits* or *Controls*. *Start* will take the player to the *World Map* screen; which includes the *Shop* and the *Level Selection*.

Each level will have right and left buttons to move the hero, attack and magic button.

The next figure illustrate how will be the user interface:

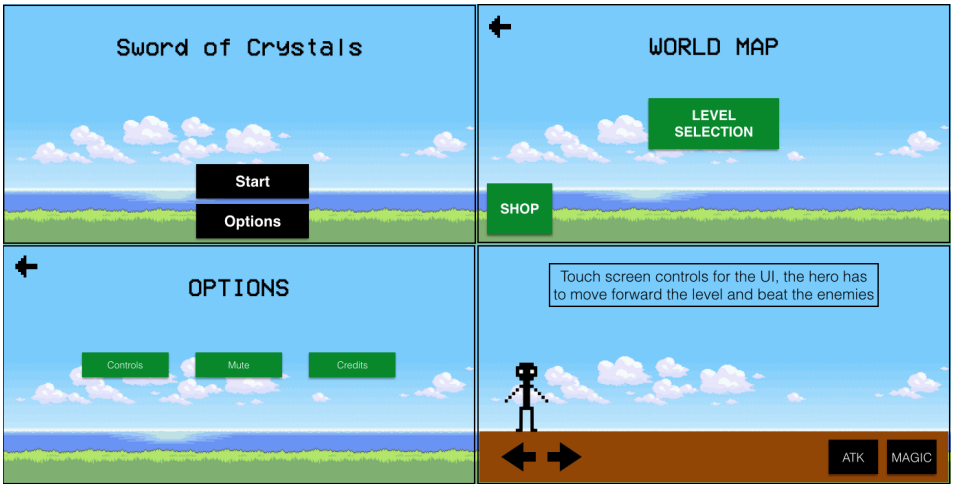



Figure 3. Prototype of the GUI

### 3.3.8. Weapons and armours

The armours would only change the appearance of the main character, in the next table they are detailed:

| Armour | Description | Price |
|---|---|---|
| Scale armor | Rusty scale armor | Initial |
| Thief suit | Light armour for stealth missions | 100 |
| Dragon armor | Armour made with the skin of a dragon | 500 |
| Golden armor | Made full of gold only for the true lords | 1000 |
| Thug life glasses | Secret item that made the character unbeatable | Secret |

The *Thug Life Glasses* are unlocked when you beat Yggdrasil for the first time. In next table the weapons are detailed, they have attack, range and speed.

| Name | Attack | Speed | Range | Price |
|---|---|---|---|---|
| Long sword | 1 | 1 | 1 | Initial |
| Bow | 1 | 1 | Distance | 200 |
| Fire sword | 2 | 1 | 2 | 800 |
| Murasame | 3 | 3 | 2 | 1000 |
| Sword of Magic | 3 | 3 | 3 | — |

*Sword of Magic* is unlocked when the player has collected the 3 crystals and is ready to beat Yggdrasil in the final battle.
Once Yggdrasil is dead for the first time, the player will be able to switch between the *Sword of Magic* and the other weapons in the shop.

The Long Bow has special range because it can attack from a long distance but with low attack and speed, it is the only "pure" ranged weapon in the game.

---

### 3.3.9. Genre

*Sword of Magic* hasn't got a defined genre because it is a platform game that could be *Super Mario Bros* for example but it also includes Action-RPG elements like the armours, weapons or life of the character or the history. It is possible to say that this game will have a hybrid game genre.

The genre that fits better with this game is *Short plays when you are waiting the bus*. Maybe it sounds a bit strange but this kind of short-play games are growing up very fast because the popularity of the smartphones. People usually play a game while they are waiting for the bus, metro, train , etc., or simply waiting for someone.

---

### 3.3.10. Levels

During this proof-of-concept development the student decided to make only 3 different levels and the final stage, maybe in the future this project is finished by adding more features and levels. The levels will have a predominant colour so the first one will be easy, then medium and then hard.

All the levels include 3 shards of crystals (gems) that have to be collected by the hero if you want to go to the next level and 2 optional hidden chests.

- Nature level (green): First level. At the beginning there will be a short tutorial explaining the controls and how to attack and use magic. There will be 10 enemies during the level (in quantity) but something like 3 or 4 different types of them and few wooden spikes.

- Fire level (red): Second level. Medium difficulty, 15 enemies in quantity and 5 or 6 different types of them. There will be flying enemies, hidden traps like spikes or fireballs.

- Death level (black): Third level. Hard difficulty, 20 enemies in quantity and 7 types of them. More hidden traps than the other levels and stronger enemies like skeletons or zombies.

- Final Battle: This is the last battle between the hero and Yggdrasil, no monsters or traps here. There will be only a battle *face to face*. Yggdrasil will have a health bar on the top and he is able to attack very fast and use ranged or magic attacks. Max difficulty.

---

### 3.3.11. Extras

The last subquestion of the research question is: *If the game is finally published, can I monetise it?*

If there is enough time one new feature may be added, this is de ads, like Google ads or Unity ads. Unity ads seem to be the best option though (because is a native feature).
On the other hand there is another feature, the achievements in google play games (or Game Center on iOS). These achievements can be for example: "*beat 100 enemies*", "*beat Yggdrasil" or "buy Murasame".*

## 3.4. Implementation phase

The implementation phase is the moment when the student will try to create the game according with the GDD previously written and the conception of the game in his mind. After reading and watching tutorials about the tools that are needed for the development, the first step is to create some sprites and then start programming with them. The graphic section requires a huge load of work and thats why the student should start early with them.

The approaching followed by the student is to create a *test level* and implement all the features like traps, enemies or crystals. The reason of doing that is because this *test level* doesn't require many sprites to start programming the AI or CD.

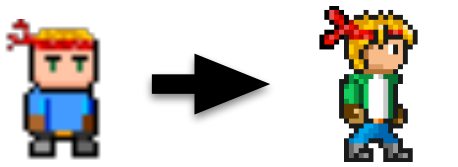
The second step is to draw the remaining sprites and create the full playable level with all the features created before in the *test level* and finally create the remaining levels, the touch screen UI and the GUI.

---

### 3.4.1. Graphics and animations

The graphic technique of game is called pixel art which is very popular actually because gives *retro* style to the game. This technique consists in creating the assets on the pixel level using for example in this project Adobe Photoshop CC 2014. The size of the pencil is 1px and it is mandatory to disable all the anti-aliasing or blurring filter because this filter calculates new pixel values automatically.

The first sprite drew was *Elkyy*, without using any scanned image to draw over. The character is inspired in *Locke* from *Final Fantasy VI*. There was a first approach with an early 8 bit version. After this first approach *Elkyy* was modified to have a 16 bit appearance which is more colourful and detailed. Once the hero was created there was the time to create some background to use in the *test level*. To create backgrounds or big structures is very common to draw a set of for example, 8 tiles with a 32x32 size and then create them like if it was a puzzle, for the test background there were 5 different tiles.



First approach
(8 bits)

Final approach
(16 bits)

Figure 4. Drafts of Elkyy

Creating the enemies and the traps the same process was followed, drawing them without using scanned images, directly in Photoshop. There are three kind of enemies, the slime, the turret and *Yggdrasil*. Each one have approximately 10 different colours. When using a limited palette it is usually to use a technique called *dithering* to achieve different colours or shades, *dithering* was used often in the assets, specially in the backgrounds.

The hardest part of the graphic design was to create the first level. There is a tool called *Tiled2Unity* where you can use your tiles to create a big map and then export them to Unity with all the colliders included, the problem was that this plugin only works in Windows and the student was using Mac. So, with a set of 60 tiles of 32x32 each the first level was created *handmade* using Photoshop. It took about 6 hours of working but the result was good, the size of the map is 2134x834.

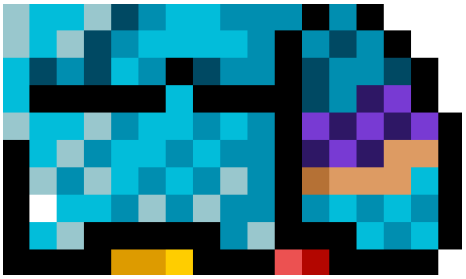


Figure 5. Dithering on Yggdrasil

After creating a sprite in order to import to Unity it has to be saved in PNG format and in Unity use 18 pixels per unit without filter mode and Truecolor format.

The animations in 2D are a *tough* process. Every animation is a sequence of a 4 or 5 different sprites everyone handmade almost from 0. For example, the next figure shows the animation of *Elkyy* walking:

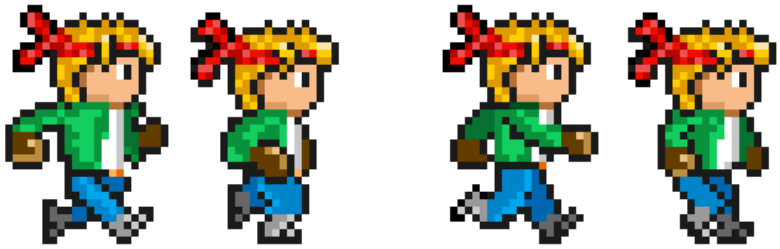


Figure 6. Walking animation of Elkyy

The only useful part of every sprite was the head, the rest of the body was completely redesigned. It is very important to make all the sprites with the same size, doesn't matter if most of the image is only background. This is because when using the sprite editor from Unity it will slice automatically the sequence of images in different sizes which means different pivots (centres). If the animation is composed with images with pivots very distant between them, the animation won't be good. Luckily for hard animations like *Yggdrasil* attacking with his sword there is a little plugin called *PivotEditor* which can centre automatically the pivots in images with huge movements or variations.

Unity has his own *Animator* so the student used it instead of software from third parties. The *Animator* consists in a finish state machine switching between states if the conditions are meet. Normally each animation runs under 60 samples (FPS) with durations between 0.5 - 1

seconds. There are special animations like the chest opening running at 10 samples to have a smoother movements. The next figure shows the finish state machine of *Elkyy:*

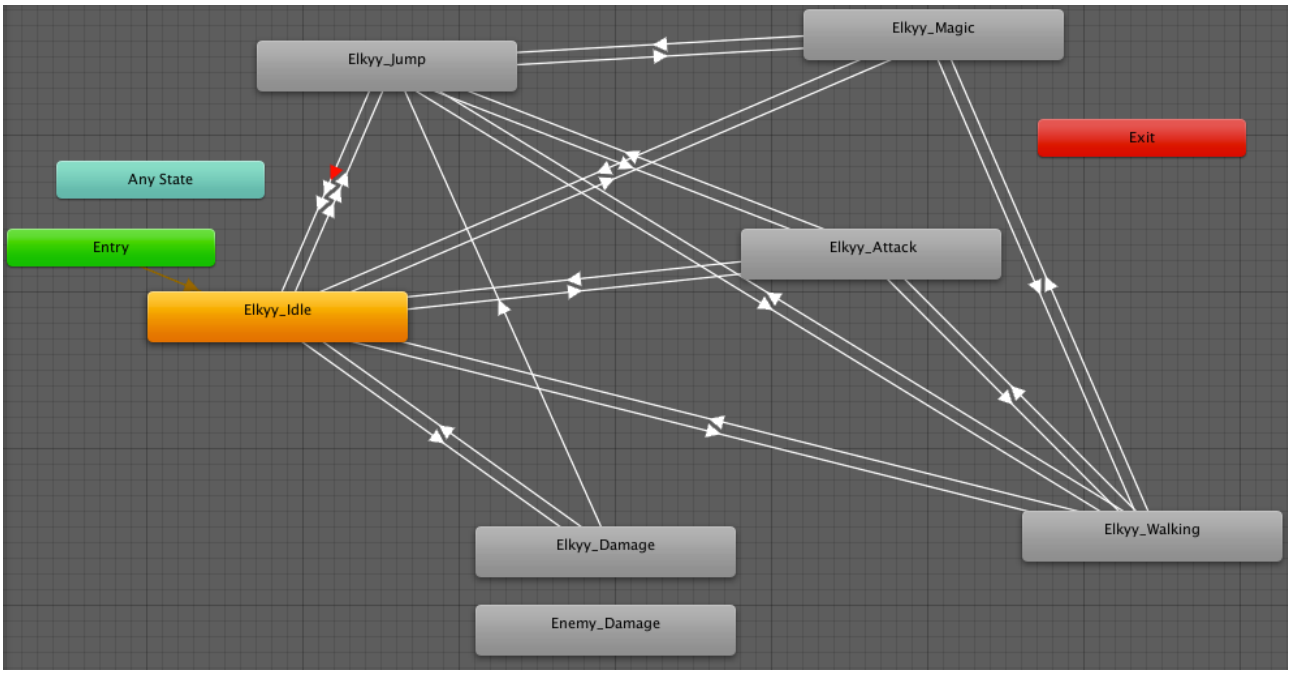



Figure 7. Animator of Elkyy

### 3.4.2. Programming

The programming part has been done with Unity and MonoDevelop (his own text editor). It is possible to use Visual Studio instead but the student worked already with MonoDevelop and he felt good using it again. Unity has a large community with forums and support in C#, so all the project was done with it. Another vantage of using Unity is that is an environment with useful tools and libraries that make easier to start into the game industry for the newbies. With only one code it is possible to port the game to a lot of platform like Android, iOS, PS4, etc., this feature can reduce the workload of porting games massively.

The programming of the game was divided in several parts:

The first part was to create and control the behaviour of *Elkyy.* Actions such idle, walking, damaged or attacking were created. To control these behaviours it is necessary to modify the variables of the *Animator*.

The instruction *SetBool* is needed to modify these variables and switch between states. In this way when *Elkyy* is standing idle the variables are *grounded* and *speed* less than 0.1 so if the player hits the attack button the *SetBool* instruction will modify the variables and then switch the state.

*Elkyy* is a Game Object, this means that has 7 different types of components like the sprite renderer, animator, box collider 2D, some scripts attached and the most important part, the *RigidBody 2D*. This component places an object under the control of physics, thus the object will have mass, gravity, etc. The values of the parameters used are the given by Unity but in 2D games is necessary to mark *Freeze rotation* in axis z.

There are sub components (or child):

- Groundcheck & Wallcheck: Two colliders with *is Trigger* activated to control when the player is grounded or near a wall in order to not fall or pass through.
- AttackTrigger: Initially disabled this collider is the area of the attack.
- TargetMagicPoint and MagicPoint: These two colliders are used to shoot the magic ball from the hero in a straight line. MagicPoint is where the fireball object is created and TargetMagicPoint is the direction followed by him. One way to program this behaviour is using the function *Instantiate* as GameObject to create a different object every time the player hit the button magic.

The behaviour of these colliders is controller by scripts and the function *OnTriggerEnter2D*, *OnTriggerStay2D* and *OnTriggerExit2D*. These functions do something when the collider is triggered by other element in the level.


Figure 8. Main screen of Sword of Crystals

Once the behaviour of the player was created, the enemies had a lot in common in the programming part.

- Turret: This enemy is hidden under the ground and when the player approaches it has two *Polygon Colliders 2D* wider in the end to detect if the player is close enough. If it is the turret appears and shoot a fireball in the direction of the player. This direction is defined by a *Vector2 direction* composed by the position of the target minus the position of the turret. His max health is 3.
- Slime: The easiest enemy, slime is moving in a fixed path all the time. To achieve this it was necessary to create a game object including the slime and two colliders along his path. When the slime trigger a collider (left or right) changes his direction to the right or to the left using *transform.localScale* and switching the value of x between 1 and -1.
- Yggdrasil: This enemy is supposed to be the hardest to beat, he has components in common with *Elkyy* like the *AttackTrigger* or the *MagicPoint* but in this case there is an

extra component called *AttackCone*. This component is a collider 2D with *isTrigger* enabled and his function is to detect when the player is inside his range and then attack with the sword dealing 1 point of damage. He also has two more *isTrigger* colliders in the edge of the map to change the direction when trigger like the slime. When his direction changes he shoot a fireball in a straight line towards the hero. Finally there is a life bar showing the hit points remaining, starting with 6.

This is everything concerning the AI of the game, the next step in the programming tasks was to create items to help the player. These items are essentially sprites with a collider 2D *isTrigger* and a script to add the behaviour. The items were:

- Coin: Needed to buy items in the shop (not implemented yet), when the player goes through it disappears and add 1 to the coin counter of the HUD.
- Purple coin: Same behaviour as the normal coin but this one adds 10 to the coin counter.
- Heart: Restores 1 health to the player.
- Mana potion: Restores 1 magic point to the player.
- Chest: It can drop variable items depending of the value of a random float number. If the value is greater than 0.5 the drop is a normal coin, if it is between 0.30 - 0.49 the drop is a purple coin and finally if the number is less than 0.29 the drop will be a heart.
- Gems: The *soul* of the game, the player needs to collect the three gems in order unlock the boss level. If the player collect one gem there is a *PlayerPref.SetBool* to save it and it is not necessary to collect again the same gem, the value is saved.

In the first level there are only normal coins spread in the map and two chests with random drop but also the enemies can drop items following the same procedure as the chest but in this case, they can drop a mana potion as well. And then, in the boss level, there are two hearts and one mana potion to help the player to beat *Yggdrasil* easily.

The last part of programming the levels and gameplay was to implement a trap *spikes* which deal 1 damage to the player and activate a custom function called *Knockback*. *Knockback* adds a force to the player in the x and y axis to beat back the character. The other important element are the platforms that the player can jump from below. This feature is new in Unity and it is a kind of collider called *Platform Effector 2D*.


Figure 9. Touch controls

Last but not least was the programming of the GUI with touch screen controls. There is a HUD that shows the health of the player, magic points, coins earned and the touch controls with a left / right arrow and jump, attack, magic buttons.

All these elements are inside a *Canvas* and controller by a *GameMasterController* which reduces the health or mana if necessary or add coins. There is also a *Pause Menu* that allows the player to reset, restart the level, quit the game or go back to the main menu. Unity implemented a UI system since 4.6 that makes really easy and fast to create those menus. The function *onClick()* allows the button to execute any function of a script attached to an object.

The flow of the GUI is the following:

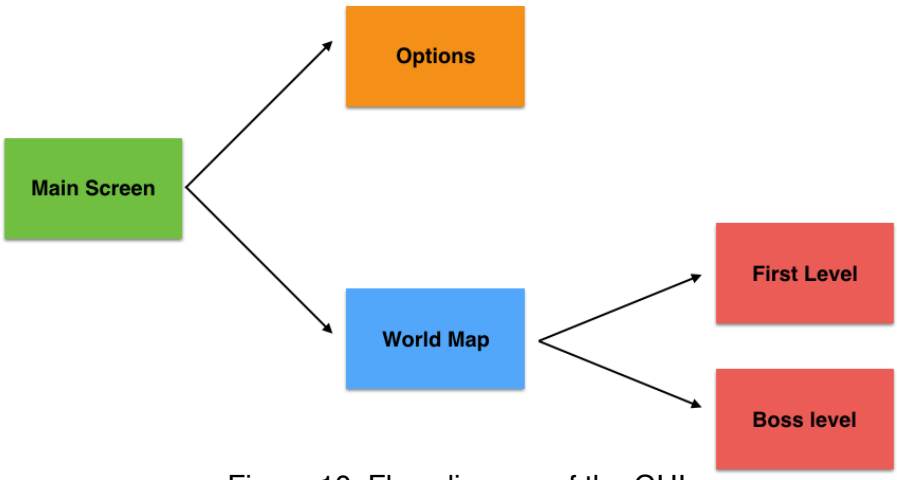



Figure 10. Flow diagram of the GUI

In the Main Screen the player can go to the options menu and erase all the data, which means that all the *PlayerPrefs* saved are deleted. The World Map shows up the first level with three gem sockets, if the three sockets are full, Yggdrasil level is unlocked otherwise an error panel will show up telling the player that is necessary to collect the three gems first.

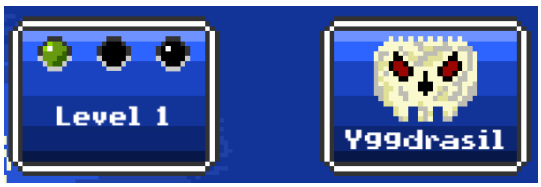



Figure 11. Touch buttons of the two levels

The error panels are set as *GameObject* and *SetActive(false)* in the *Awake()* function (this means that are disabled before the *Start()* function is called). If there is an error then the state changes to *true* and the player can see the error. By pressing *OK* the state comes back to *false* and the panel is gone again.

## 3.5. Testing

Testing is one of the most important part in the game development process. By testing the game the developers get feedback from the testers or normal players about existing bugs or features that are not as good as expected.

The first approach for testing a game is doing it by modules in the implementation phase, this means that the student tested for example the turret hiding/rising or detecting the player were working correctly separately and then when creating the turret the modules had

the same behaviour. There were errors putting together the slime and his colliders (which make the slime to change the direction). The solution was to create a parent *GameObject* and making the two colliders and the slime itself as child.

Once the module and integration testing was a success the student tested the whole game to ensure that there weren't major bugs in the game. The device used in the test was a Xiaomi Mi3 running under MIUI 7 (Android 5.0).

Porting games from Unity to mobile devices sometimes could be hard because of the resolution of the screen. There are a lot of different devices running different versions of Android or different screen resolutions. The default orientation of the game was set to *Landscape Right* and the minimum API level to Android 4.4 Kit Kat. The result of the testing was pretty good with minor errors though. These errors were for example with the platforms and the new feature of Unity *Platform Effector*. Sometimes the character is not passing through the platform correctly for example. Other errors where with *Yggdrasil* and the collider that detects the player, sometimes the collider wasn't acting well. This is mainly because Unity was made originally for 3D games and sometimes the 2D features don't behave as expected but with every new version they release some problems are fixed.

Finally 3 different players tested the game in his Android devices and the result was quite good. They found some errors with the menus that were fixed, for example, if the player collected the gems the *PlayerPrefs* weren't acting correctly. Thanks to his feedback minor bugs were found and fixed correctly also they feel comfortable with the position of the touch screen controls and the GUI.

# 4. RESULTS AND CONCLUSIONS

## 4.1. Project management

Creating a game is not an easy process, as mentioned above is an iterative process where the game can change his features a lot of times. Being a team of one person means that the student had to assume all the roles (i.e: game designer, programmer, art) with no previous experience but managed to create a complete game design document with all kind of details although not all the features are implemented in the actual game.

Planning the game is a hard aspect to lead because the student had to define milestones and tasks without knowing if X task consumes more time than Y for example. Thus, some aspects of the GDD or in the project plan could be not very realistic for the time needed to do all the things planned. Clearly the time was the worst impediment when doing the project due to the lack of experience of the student. In the GDD the game had 3 levels + boss level but with only 4 months of developing the game actually is 1 level (a mixing of the 3) + boss level. This is not necessarily bad because the student learnt the true effort and resources needed when creating a game.

Despite these bad aspects in the planning, the management of the project made the student participate in a real development process writing and thinking each aspect of the game like he can do in the future in a real company.

## 4.2. Game implementation

During the implementation the student encountered a lot of difficulties especially in the beginning. Doing all the assets requires a huge amount of time that were not expected in the project plan, pixel art is beautiful and gives a *retro* aspect to a game but is very laborious.

As mentioned above, there are aspects of the project plan or the GDD that are *unrealistic* and were modified during the implementation through iterations and modified according to the time and the resources available. Implementing only 1 level + boss level was a good idea because it gave the student enough time to work thoroughly with the less amount of bugs possible with a good result. There were other problems like the plugin *Tiled2Unity* doesn't work in MAC so the map was drawn without any help and consumed almost 1 day of work to simply draw it.

In general all the visual design took almost 50% of all the time available in the project, it is ambitious for a bachelor student to create in his first approach a game with a good pixel art design but the result is still good. The student preferred to draw every pixel of the game than using existing assets in Internet this means:

- More than 40 .PNG files with handmade sprites
- 19 .PNG for the animations which means that each .PNG includes 4 or more sprites needed in the animation process
- Motivation of the student to create his own game without any external help, every part of the game is totally of his property, maybe in the future he would complete the game and try to publish it.

The programming part wasn't easy but the student had a previous background so this part of the implementation consisted in reading manuals and tutorials about creation of 2D games in Unity and then implementing some features like the AI, GUI or CD.

Finally, the testing is divided in two parts, the first one is the testing by the student at first by modules and then the integration of everything. This part of the testing had been done at the same time of the programming and drawing in the beginning, in the *test level*. So the consumption of time is not noticeable because it was part of the programming part. The second part is the public testing by 3 different real players in 3 different devices, it was good because received real feedback about the game. This feedback helped to fix bugs and to have a real impression of the game, for example if it was *good idea* or *addictive.*

## 4.3. Research questions

This section will be about the early objectives and the real result achieved. The early features were good theoretically but when the student tried to do it was not possible mainly for the lack of time and experience.

The result was good though because the sub question: *How games are designed, developed and tested?* The student learned step by step the whole process of doing a game. From designing the GDD, then implementing it and finally the testing part. He learned that the initial idea

or conception of a game won't be the final result, the process is changing as the implementation is in course but this is good because in real studios the same thing happen, it is always an iterative process.

There is another sub question: *Can I use my previous knowledge in signal processing to create efficient graphics and audio? How my tools will match in the game development world.* The student was familiarised with image and audio processing tools like Matlab and he had extensive background in audiovisual signal processing so he use it for example to determine that the .PNG format was the best option for saving the sprites. The final answer to this question is that yes, the student used his previous knowledge to create efficient assets but it's not mandatory to have this previous knowledge though.

The most important question of the whole project was: *Is it hard to a student without previous experience to get into the world of games? Will I be capable to learn how to use the different tools and software involved in a game development process?*

The answer again is yes, a single student without previous experience *can* develop a game but it has to be a realistic project and the time needed to do it is also big. Developing a game assuming all the roles is a tough work, the student has to read a lot of manuals, books and tutorials to acquire some knowledge in all the fields, i.e: level creating, sound, art, programming, business, etc. So the first approaching to the game industry based on the results of this project has to be a *simple* game with reachable features or objectives. The GDD of this game was very extensive therefore not realistic with the time available and there were features not implemented or others trimmed like the 3 levels became 1 level. With more time and students in the team like 1-2 programmers, 1-2 artists and 1 game designer the result will be much better and almost professional but with a single student the result is pretty good but not *professional*.

Despite that, the student is happy with the final game, it could be useful to show as port folio when applying for a real game job or the game can be fully developed and published in the future.

# 5. FUTURE WORK

The future work means completing the game according to the GDD with all the features thought by the student in the beginning. It means a redesigning of some assets and making the code more efficient. This will be possible because the student will be more experienced in the future or would have external help like professional artists or sound engineers.

The next table shows the features not implemented yet:

| Feature | Explanation | Reason |
|---|---|---|
| Shop | The shop is a NPC included in the world map that allows the player to buy weapons or armours and change the appearance or power up the character | The shop is not implemented because it requires a lot of new sprites to work. The programming is not hard but the art is a huge work |
| Sound | Sound effects and soundtracks for the game | Lack of time mainly |
| Levels | 3 levels mentioned in the GDD, one for each element of the world | Drawing the sprites and assets needed for a single level took about 3 weeks and programming the logic part 2 weeks. Creating 3 different levels is possible but much more time is needed |
| Introduction & tutorial | Brief introduction telling the story of the game and tutorial to explain how to play | Same reason as the levels, the time was very limited and doing this part requires a lot of new assets |

All these features will be implemented and completed if the student decide to finish the development of the game in the future.

The last feature to be added in the future would be the *Ads*. Nowadays there are *millions* of apps and games available so the best option to promote the app and attract users to download your game is release it free.

The best options for the ads is Unity Ads because is a feature that has full integration with Unity3D and becoming very popular in the last months. Companies like SuperCell, Eight Pixel Square or Mag Interactive use it with successful results. *Crossy Road* generated $3 million in a few months [8], this means that the future of the industry or at least the future of smartphones games is to be released under a free model implementing ads.

## 5.1. Publishing the game

Everybody wants to see their projects published and making profit with it, after all, creating games is a business. In this section some details about publishing a game will be commented, focusing in iOS and Android because this project is about a portable devices game.

These are the requirements to upload a game to the Play Store (Android) [6]:

- Pay a $25 registration fee (once).
- Rate the maturity of your app.
- Google only allows apps of 50MB or less for a single APK, if the app exceeds this size need to use the Android's APK expansion files API which allows to store an additional 4GB data in the Google's servers.

One of the most important thing is the listing details of the app, this element will help to promote it. So its needed:

- 2 screenshots at least but for best results the developer should upload 8 screenshots and if wanted the "Designed for tablets" list it is required to upload a screenshot of the app in a 7-inch or 10-inch device.
- Hi-res icon (512x512), Feature graphic (1024x500) and Promo graphic (180x120).

Finally if the developer want to do a device filtering there are two options:

- Device availability provides a dynamic list of supported devices based on the Android Manifest of the app. For example, if the manifest says that the screen resolution is for large screen size, the console will reflect the supported devices that can find the app in Google Play. [7]
- The developer can filter non-compatible devices manually by adding those that are legacy or presented problems during the testings. If a device is in that list, it won't be available in Play Store for them.

This is all the information related to the Play Store, now these details for publishing in App Store (iOS) [9]:

- Pay $99 fee every year.
- App's icon sized 512x512.
- At least one screenshot sized 320x460.
- Choose the availability date for the app. This can be as soon as the app is approved by Apple, a certain date in the future chosen by the developer but it may not be approved on time and then it will be released as soon as approved like the first option. And finally the developer can choose the release date once the app has been approved.

- Choose the price tier or release it free. The developer can mark the option to offer discounts to educational institutions.

# 6. REFERENCES


[1] *The evolution of video gaming and content consumption,* 2012. Available from: <https://www.pwc.com/us/en/industry/entertainment-media/publications/assets/pwc-video-gaming-and-content-consumption.pdf>

[2] ME Moore (2011), *Basics of game design,* Boca Raton, FL: CRC Press, pp. 7-11

[3] Blow, J., *Game Development: Harder than you think,* Available from:< http://queue.acm.org/detail.cfm?id=971590>, February 2004

[4] Kasurinen, J., Mirzaeifar, S. and Nikula, U., 2013. *Computer Science Students Making Games: A Study on Skill Gaps and Requirement",* Procs. 13th Koli Calling International Conference on Computing Education Research , New York, NY, pp. 33-41

[5] Kasurinen, J., Strandén, J. and Smolander K., 2013. *What do Game Developers Expect from Development and Design Tools?,* Proc. 17th International Conference on Evaluation and Assessment in Software Engineering (EASE), 14. – 16.04.2013, Porto de Galinhas, Brazil.

[6] "Android Authority", [online], http://www.androidauthority.com/publishing-first-app-play-store-need-know-383572/, [accessed: January 2016]

[7] "YoYo Games", [online], http://help.yoyogames.com/entries/21959497-How-do-I-publish-on-Google-Play-, [accessed: January 2016]

[8] "Pocket Gamer", [online], http://www.pocketgamer.biz/news/60900/crossy-road-3-million-unity-ads/, [accessed: January 2016]

[9] "Ray Wenderlich", [online], http://www.raywenderlich.com/8045/how-to-submit-your-app-to-apple-from-no-account-to-app-store-part-2, [accessed: January 2016]

[10] "Serious Games Network", [online], http://seriousgamesnet.eu/assets/view/238, [accessed: October 2015]