

# Helping activate children through the use of video games

—  
**Jørn Vollan Lomax**

*INF-3981 Master's Thesis in Computer Science - June 2015*





# Abstract

The video games industry is now one of the biggest entertainment industries in the world. Forbes magazine estimates that the video game industry will sell games for 70 billion dollars by the end of 2015, and the biggest growth is in the mobile market. While most of the video game industry is creating games strictly for entertainment purposes, there is a growing demand for games that can be used for other applications. This paper will look into making games that help children learn and prepare them for the highly technological world they will grow up in. Touch screen were a novelty 10 years ago, now touch screens are found in multiple places, and it can be beneficial to introduce children to this technology early. The paper will also look at how video games are created, the limitations of creating games for mobile devices and mobile technology and look at some of the benefits and drawback of mobile games.



# Acknowledgements

I would like to thank Ismet and everyone at PlusPoint. You have given me lots of enthusiasm and pushed me to pursue my dream of developing computer games.

I would also like to thank my classmates for being a good support and a major distraction. Without them I would have released a second game by now, but it would not be nearly as fun.

And finally I would to thank my wife for putting up with me for the past 5 months. Now i can finally go back to husband mode.

*If you try and take a cat apart to see how it works,  
the first thing you have on your hands is a nonworking cat  
-Douglas Adams*



# Contents

|          |                                     |           |
|----------|-------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                 | <b>1</b>  |
| 1.1      | Goals . . . . .                     | 1         |
| 1.1.1    | Elaboration . . . . .               | 2         |
| 1.1.2    | Limitations . . . . .               | 3         |
| 1.2      | Thesis Summary . . . . .            | 3         |
| <b>2</b> | <b>Designing a game</b>             | <b>5</b>  |
| 2.1      | Computer gaming . . . . .           | 5         |
| 2.2      | Creating a game . . . . .           | 9         |
| 2.3      | Technical background . . . . .      | 11        |
| 2.3.1    | Bluetooth Low Energy . . . . .      | 12        |
| 2.3.2    | Game engine and Libraries . . . . . | 12        |
| 2.4      | Summary . . . . .                   | 13        |
| <b>3</b> | <b>Conceptualizing the game</b>     | <b>15</b> |
| 3.1      | Concepts . . . . .                  | 15        |
| 3.1.1    | Beacons . . . . .                   | 15        |
| 3.1.2    | Guardians . . . . .                 | 16        |
| 3.2      | Server . . . . .                    | 16        |
| 3.3      | High level concept . . . . .        | 17        |
| 3.4      | The Game Design Document . . . . .  | 17        |
| 3.4.1    | Gameplay . . . . .                  | 18        |

---

|          |  |           |
|----------|--|-----------|
| 3.5      | Alternatives . . . . .                   | 19        |
| 3.6      | Summary . . . . .                        | 20        |
| <b>4</b> | <b>Implementation</b>                    | <b>21</b> |
| 4.1      | Game engine . . . . .                    | 22        |
| 4.2      | Middleware/Backend server . . . . .      | 22        |
| 4.2.1    | Parse.com . . . . .                      | 24        |
| 4.2.2    | Other possible server options . . . . .  | 25        |
| 4.3      | The game client . . . . .                | 25        |
| 4.3.1    | Saving and loading data . . . . .        | 26        |
| 4.4      | Summary . . . . .                        | 27        |
| <b>5</b> | <b>Reviewing the work</b>                | <b>29</b> |
| 5.1      | The final game . . . . .                 | 29        |
| 5.2      | Scenes . . . . .                         | 29        |
| 5.2.1    | The main Menu . . . . .                  | 30        |
| 5.2.2    | Guardian scene . . . . .                 | 32        |
| 5.2.3    | Battle scene . . . . .                   | 33        |
| 5.3      | Testing . . . . .                        | 34        |
| 5.4      | What could be done differently . . . . . | 35        |
| 5.4.1    | Alternative technologies . . . . .       | 35        |
| 5.5      | Future work . . . . .                    | 36        |
| <b>6</b> | <b>Conclusion</b>                        | <b>39</b> |
|          | <b>Appendices</b>                        | <b>41</b> |
|          | <b>Appendix A Game Design Document</b>   | <b>41</b> |
|          | <b>Appendix B High Level concept</b>     | <b>55</b> |



# Acronyms

**BLE** Bluetooth Low Energy.

**DDR** Dance Dance Revolution.

**FPS** First Person Shooter.

**MMORPG** Massively Multiplayer Online Role Playing Game.

**MUD** Multi User Dungeon See: 1, *Glossary*: MUD

**NDK** Native Development Kit.

**RPG** Role Playing Game.



# Glossary

**MUD** Multi user dungeon. A multiplayer RPG game where many players can play together and interact.

**Multiplayer** A game where two or more players play within the same game. This can be either online, or on the same computer.

**Rouglelike** A subgenre of role playing games. Most often characterized by procedurally generated dungeons, turn based gameplay and permanent death. Name comes from the game "Rouge", which was released around 1980. Modern day examples include the Diablo series, "The Binding of Isaac" and "Don't starve".

**Stats** The data representing particular aspects of a fictional character. It defines what their attributes are.

**Story bible** A document describing the history of the universe within the game. It explains the reasoning and thought behind the current events in the game and any thoughts of the main characters.

**Tween** Short for Inbetweening. It creates a very fluid animation by making sure an animation is handled neatly in between key frames .



# List of Figures

|     |   |    |
|-----|---|----|
| 2.1 | The very first version of Tetris . . . . .                      | 7  |
| 2.2 | Bartle graph . . . . .  | 8  |
| 2.3 | All derivative version of the original quake engine . . . . .   | 13 |
| 4.1 | Losses carry more value[...] . . . . .                          | 26 |
| 5.1 | The registration screen . . . . .                               | 30 |
| 5.2 | Program flow for the character name creator . . . . .           | 31 |
| 5.3 | The main screen . . . . .                                       | 32 |
| 5.4 | The available backgrounds for the guardians placed on the map . | 33 |
| 5.5 | Early version of the battle scene . . . . .                     | 33 |



# List of Tables

|     |  |    |
|-----|--|----|
| 4.1 | Comparison of different libraries/engines to create games with . . | 22 |
| 4.2 | Middleware comparison table . . . . .                              | 24 |





# Chapter 1

## Introduction

Child obesity has become an increasing problem over the past decade. A US study which looking at child obesity from 1999-2010 saw a significant rise in weight problems among children [19]. While the study made no conclusion as to the reason, it did mention that there has been a change to more sugary diets and decreased activity among children. This project sets out to help with one of those issues, and help children become more active with the help of video games. While the average age of a video gamer is estimated to be 31 years old [4], 30% are estimated to be under the age of 18. Casual/social game is the most common genre of game, and has 46% market share on mobile games. These are the facts this project builds upon, using the casual/social genre on mobile devices to help children become more active.

The way this project will solve this is by creating a game that augments the world around the player. It will allow players to explore the world around while at the same time interacting with other players. This will be an interactive and engaging experience for players. The game will use state of the art technology to help achieve this and at its heart the game is about connecting players in new ways, while also giving players a challenge that is not just in the game, but also in the physical world.

The project aims to create a game that will be engaging to children. If the game is engaging enough, so that they will want to play it by themselves and will play as a matter of choice and not because they are pushed to do so. This will be done through simple gameplay, a fun art design and giving a sense of excitement to the young players.

### 1.1 Goals

The goal of the project is to create a game that can make children more active outside of their living room, preferably outside. Children are less and less active, and child obesity is on the rise, especially in developed nations [19]. There are

games that are created to be a more interactive experience, examples include Dance Dance Revolution and Wii sports. There is however not many that also want to take the experience out of the living room. Another game that is a big inspiration for the project is Ingress. Ingress is a GPS based game where the player has to fight for territory using portals that are placed in a virtual world that is based on real world coordinates. It is this type of game this project sets out to create. The main issue with Ingress is that it is quite complex and can be hard for children to understand. It also requires the ability to move over large distances, with the aid of cars, planes or even boats, all of which are not available to children. This project wants to create a game that is mobile and can augment the world around the player, changing the context of the environment and be simple enough for children to grasp. It should excite the player and make the player want to play it more, which then forces the player to be active.

The end goal is to create a video game from scratch, which has the potential to activate children, through augmenting the world around them. The target age group is 10-14, since the game will require access to a smartphone or tablet. The video game should be in a state where it is complete and ready to be released to market by the end of the project. The project will follow the development from the very beginning, conceptualising the game, designing the game and finally developing the game. The project will not focus on any particular point of the process, but will reflect the amount of work required for each stage in the process.

The goals can be summed up with the following statements:

- Create an interactive game that is aimed at children aged 10-14.
- Create a game that forces the player to move outside to play. It should not be possible to just play from the couch.

### 1.1.1 Elaboration

The project will also require work on elements that are not a direct part of the game world. There will be need for databases and a way to administrate the game. This will require either a database server written from scratch, or the use of a 3rd party service.

The project is developed using agile development principles of; working software over comprehensive documentation and responding to change rather than following a plan. This implies that the plan originally laid out can and will change as time goes by. This is especially important in game development, where a gameplay element can look good on paper, but not be as fun as intended when tested with users.

There are some terms that are used throughout the document that are a little different from regular software development. The users of the game will often be referred to as players and the terms will be used interchangeably. "Video games", "computer games" or just "games" will also have the same meaning throughout the text unless stated otherwise.

### 1.1.2 Limitations

The project has some limitations. There are some features that would add to the game and help make it more interactive, but are not essential for the game. One example is chat amongst players. Allowing players to chat with each other will help create a sense of community among players and also help create teamwork. This project is not about chat services, so this will not be part of the project.

Since the game is written by one individual, with limited artistic talent, the graphical assets are not a priority. There will be very simple graphics. With time, it would be great to have custom art created by professional artists and/or designers. There will be a description of the required assets, but this do not have to be completed as part of the project. Music and Sound assets will also be described, but will not be a goal for the project.

## 1.2 Thesis Summary

This project will show that it is possible to create a relatively complex game within six months. When evaluated it will also show that there is a big potential for games to help children have games that are both innovative, fun and also help tackle the decreased level of activity among children.

Chapter 2 will describe how video games are designed and chapter 3 will show how this is used in the design for this project. Chapter 4 outlines the technology that was used for the game, and how it was implemented. Chapter 5 will show the results of the project and discuss what could have been done differently. Chapter 6 will conclude the project.



## Chapter 2

# Designing a game

This chapter will look at what goes into creating a game. It will detail what makes a game feel like a game, and what makes players want to play. It will look into some of the history of video games and see if it is possible to learn anything from the work that has already been done in the field. It will also look at some of the technologies that are used and why they are beneficial to creating a mobile game.

### 2.1 Computer gaming

Before describing what the game for this project is, let's look at what a video game is defined as. Looking in the Collins English dictionary from 1963 there is no definition of video game, but the word game is defined as:

**Game** *n.* diversion, pastime, jest; contest for amusement; scheme, plan of action[...]

A newer dictionary (Oxford Dictionary, third edition; 2010) has the following description of what a video game is:

**Video game** *noun* a game played by electronically manipulating images produced by a computer program on a television screen or display.

Combining these two definitions, it is possible to conclude that a video game is something that includes images on a screen and can be a diversion and a pastime. The word "contest" is also relevant. A contest can be a competition with another person, which would for example be a multiplayer game (pong is one example), a game where players compete directly with each other. The contest can also be without the game being a multiplayer game, a game where a score is calculated which can be compared to other players (Tetris is one example of this). There

can also be a contest with the game world. In traditional Roulgelelike games, the contest is with the game itself. It is about defeating the the next dungeon boss, or improving a previous score. All video games have some form of contest, against other players, against own scores or against challenges created by the game developer.

One integral part of game psychology is the risk-reward theory. Risk-reward is based on the concept that the player should be able to have choice that has some element of risk, and there should be a corresponding reward.

One example scenario is the simple card game War. The game is a two player game, where a pack of cards is shuffled and split into to stack and each player get one half each. Each player then takes the top card from their stack and the higher card of the two wins the round and takes both card into their deck. The game is over when one player has all the cards. At its base, the game only has a slight element of risk. Either the card you have is higher or lower, but the player does not have a choice so perceived risk is lower. If each player suddenly has the choice to choose one random card from their deck instead of picking the top one there is an added element of risk and potential reward. They can play the card that is usually played (the top card), or they can take a gamble that a random card will be better and receive a higher reward. The reward is getting one step closer to the end goal of the game, owning all the card in the deck.

This can then be improved on further by changing the rules a little more. If a player wins a hand by playing a random card, they gain three cards instead of one. Likewise, if they play a random card and loose, they have to give up three cards. Now there is a higher risk in using a random card, but the potential reward is also higher. In many ways, risk reward systems in computer games can be compared to gambling [1].

It is important that the reward is rewarding to the player. If the challenge in the game is to collect a very high amount of one specific item, a good reward is not to give the player one more of that very item. Likewise, with loss, it has to fit the challenges. If there is a cognitive dissonance between the challenge the player is doing and the loss or reward, it will detract from the entertainment value of the challenge.

Risk can be seen as addition of Potential loss and the reward:

$$R + L = Risk$$

Reward can be seen as the equation:

$$C + E = Reward$$

Where C is the difficulty of the challenge and E is the effort required. This shows that the perceived risk can be lowered or heightened by increasing the difficulty of the challenge, or simply changing the effort required. It is natural for a player to become better at a game the more they play it, as they develop the skills to handle the challenges. This will decrease the difficulty for the player to complete them. It is therefore important to increase the difficulty as the game progresses, otherwise the reward will reduce and so will the perceived risk.

There are also different type of rewards:

**Compounding rewards:** A reward that builds up over time.

**Tangential reward:** A reward that doesn't effect the outcome of the game.

**Progress reward:** A single event that will move a story along, or change the state of the game in some way.

In a similar manner there are different types of loss:

**Ultimate loss:** Loosing the entire game. The game is over and the player has to start from the beginning.

**compounding loss:** Loosing something the player has multiple off, so the loss can build up through time. Extra lives in games a a good example. The loss becomes greater the less the player has of it.

**Tangential loss:** Loss that does effect the completion of a game. It can be missing a cut scene, or missing a valuable item.

To show an example, the game of tetris, one of the best selling game of all time, can be analysed

Tetris [11] is a very simple game that has survived for many years and is still popular today. The very first version, as seen in figure 2.1, was created by Alexey Pajitnov and released in 1984. The premise is simple, there is a continuous rain of blocks falling vertically, one at a time. The player has to place them to create a continuous horizontal line.

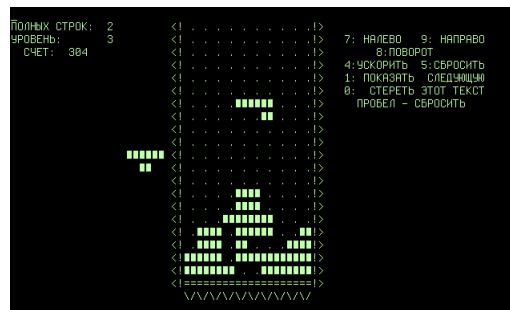


Figure 2.1: The very first version of Tetris

When a horizontal line is created, that line (and only that) disappear from the game. The Game is over when the blocks touch the top of the screen. The blocks that are falling have different shapes, so the player has to plan ahead where to place each block.

The risk-reward mechanics described earlier can be applied to a game of Tetris.

**Ultimate loss:** When any single block touches the screen, the game is over.

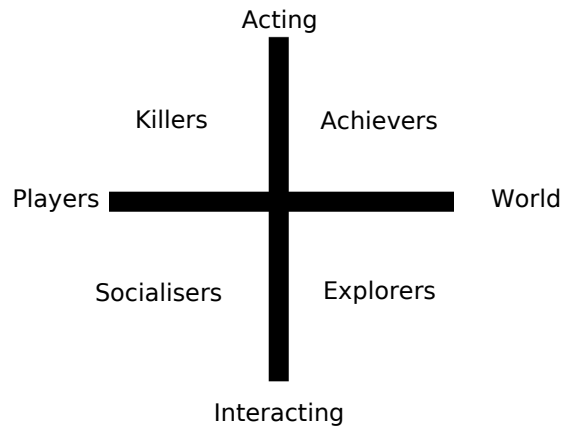


Figure 2.2: Bartle graph

**Compound loss:** When a block is placed that does not create a continuous horizontal line, the player suffers a compound loss in the amount of space available for new blocks, putting the player one step closer to an ultimate loss.

**Tangential loss:** There is no tangential loss in tetris. There are no secret levels, powerups or ending screens. There have been newer version where powerups have been added, creating a tangential loss if the player does not obtain one.

**Progress reward:** Each time the player create a horizontal line, the line disappears from the game. This gives the player a sense of progress.

**compounding reward:** A score is added up each time the player places a block and additional score is rewarded for completing a horizontal line.

**Tangential reward:** Like tangential loss, there is no tangential reward in the original game of tetris, which also like the tangential loss has been changed in some recent version of the game.

Another tool when creating a video game, especially one that is a multiplayer game, is the Bartle test. The test is based on a paper written by Richard Bartle [3] in 1996, which sets out to describe different types of players in a Multi User Dungeon (MUD) game. Although Richard Bartle has no qualifications as a psychologist, the paper has become a very popular as a way of looking to target a certain audience to a game, especially Massively Multiplayer Online Role Playing Games, and the paper has over one thousand known citations<sup>1</sup>. The test can also be applied to games that are not derivatives of mud games too. In a paper investigating how players interact within the FPS game "Counter Strike" found that players interactive in many different ways, beyond what the game originally intends [23]. These can be seen as explorers or socialisers. This

<sup>1</sup><https://scholar.google.no/scholar?q=bartle+richard>



shows an example where adding a multiplayer- (or online-) component to a game suddenly allows for new types of players to enjoy and interact with the game.

Video games have been around since the early 70s but have in recent times started to become an industry worth millions [11]. In 2005 the video game industry surpassed both the movie and music industry in term of revenue<sup>2</sup>. Video games have also entered the mainstream culture, and where video games were once seen as a "nerd" thing, it has now become very commonplace to the point that most people with access to a computer of any type have played a game [21]. Video games have also moved away from the entertainment industry and helped with other aspects of modern life. Computer games are used for training US navy pilots<sup>3</sup>, teaching children<sup>4</sup>, and activating and helping give a pastime to elderly [14, 7].

With these changes, games are more and more being used as a utility to help a person with other issues. One study looked at the effects of using the game tetris as a way of preventing soldiers of developing PTSD [9]. The study showed that playing the video game after a traumatic experience reduced the amount of flashback the user would have after the event. In another survey done among therapists and physiotherapist of burn victims, 71% believed that using a wii console twice a week would improve recovery on neurological damage and 68% believed it would help patients recover from trauma from a serious burn injury[5].

This project also has a utility value. It tries to help make children more active using computer games. There have already been studies done to see if the wii console help increase energy expenditure in children[6]. It showed that simple games like wii bowling increased energy levels by as much as double, while playing Dance Dance Revolution (DDR) on level 2, would increase energy expenditure by factor of 3.

## 2.2 Creating a game

This project sets out to create a game that has a core intention of making children more active. It tries to do this by creating a game that rewards players for exploring the wilderness. For this game it takes a starting point of using the nature in the surrounding area of Tromsø, but it can easily be expanded to work in other areas. It can be created as one big game that covers a wide area, or it can create an instance over only a small geographical area. There can even be several instances of the game running at any one time.

This ad-hoc solution allows the game to be customized to fit events, gathering or other occasions. The characters in the game can be easily be customized to, so it can even be used for advertising.

The game also takes use of Bluetooth Low Energy beacons, which allows beacons to be placed inside a building. As such, the game can be used to help familiarize

---

<sup>2</sup><http://www.gamedaily.com/articles/news/report-game-sales-blow-past-music-in-uk/?biz>

<sup>3</sup><http://www.gizmag.com/go/7167/>

<sup>4</sup><http://minecraftteacher.tumblr.com/>

employees with a new building, or new students can use it to find their way around a new school, similar to how an Easter egg hunt would work.

The game is going to target mobile platforms. Since the game requires the player to move around in the physical world around them, any other platform would be very inconvenient. Since the game will be for mobile platforms, it will be released on the Google play store and the Apple app store. Pricing is undecided, and can be revised when and if the game reaches the stage where it is possible to release it to the two markets.

The end goal for the project is to have a game that is in a publishable state that can be released to the market. It will require all components of the application to be made, both the server and client. The game should be easy to customize so that the content can be easily changed to fit the requirement of the game administrator. It should also be easy to create separate instances of the game, without any connection between them. It should be possible to create new beacons and place them from an administration panel so that new beacons can be placed for each instance. The administration panel should be specific to each instance of the game.

When looking at some of the best selling and most critically acclaimed games through history, it is very hard to see any pattern as to what makes a very good game. There is no recipe to create a game that will be both well received by critics and gamers alike. As an author's personal note, all the top rated games in the metacritic database<sup>5</sup> are, in the end, very fun games to play. It's hard to pinpoint exactly what it is that does this. The games are generally hard, but not too hard. The difficulty scales as the game progresses (Diablo, Zelda:OoT). Where applicable, they have great stories (Final Fantasy IX, Metal gear solid). They all have simple controls that are intuitive (Tekken 3, Tony Hawk Pro Skater 2). The majority of the top rated games also allows for different styles of gameplay as the game progresses (Zelda:OoT, Final Fantasy 7,8,9), through mini games or different game modes. Some were just very original games when released (minecraft, the sims, Rouge, Portal) and were the first of their kind, often spawning completely new genres of games.

All of these points are something that help to create a truly great game. It's not possible to hit all points, but they give some examples of what can elevate a good game to a truly great game in the eyes of both players and critics alike. We will look at how some of these work together when creating a video game.

Using all the information gathered about computer game theories and looking at what creates a great game, it's possible to start creating a concept for a game and applying the different theories. This will create a game that will be both fun and engaging for the player.

---

<sup>5</sup><http://www.metacritic.com/game/legacy>

## 2.3 Technical background

There are also many devices that have support for Bluetooth smart, including phones, TVs, and so called "smart sensors" like pulse monitors, cycling sensors, garage door controllers.

The beacons used to for this project are built on Bluetooth Low Energy, and can therefore last a long time without having to replace batteries. This project will focus on the features provided by BLE, but will make no distinction between BLE and Bluetooth smart.

There have been examples of games using real life location before. A patent was filed in April 2007 [20] for "Multiplayer games using GPS-enabled portable gaming devices". This was only two months before the release of the first generation of Apples iPhone<sup>6</sup> which was the first smartphone to have the capability of running games of this kind.

Patent for similar games soon followed. Two years later a patent for "Role based game play on a social network" [8]. While this patent does not involve the use of location (GPS or other methods), it does lay out a game based on a client server model where each player has a character on their mobile device.

There are also two games already in the market that serve as a inspiration to this project. The first is GeoCaching. GeoCaching is not a video game, but a recreational activity. The concept it that participants (GeoCahers) hide a waterpooof container with a log book and a pen or pencil. They may also hide a toy or small ornament. The coordinates of this cache is then written down and shared with other GeoCachers.

A GeoCacher can then enter the coordinates on their GPS enabled device. They will then try to find the hidden cache using only the coordinates and maybe a small written clue. Once they find the cache they write their names in the log book and replace the toy/ornament with a new one. The game has proved very popular with over 2500000 active caches and over 6 million participants worldwide<sup>7</sup>.

The other inspiration for this project is Ingress<sup>TM</sup>. The entire gameplay for ingress is very complicated, so it will only be described briefly. There are portals placed on a map that is based on a real geographic map. These portals can be controlled by one of any two teams at any one time. Two or more of these portals can then be tied together, creating a link between them. The team that creates the link then receives points based on the distance between the portals linked together.

There are also application that use GPS which are not games. There is the obvious application of navigation, it's most common application, it is also used for monitoring tectonics, analysing teams in sport events, geotagging pictures or placing and retrieving physical object somewhere.

---

<sup>6</sup>[http://en.wikipedia.org/wiki/IPhone\\_\(1st\\_generation\)](http://en.wikipedia.org/wiki/IPhone_(1st_generation))

<sup>7</sup><https://www.geocaching.com/?guest=1>

### 2.3.1 Bluetooth Low Energy

Bluetooth Low Energy and GPS is a technology that this project relies heavily on. GPS is a well tested and well known technology, and there is plenty of easily accessible literature on the subject. BLE is however less known, and the reader might not be familiar with the technology. BLE is a specification of Bluetooth that sets strict constraints on power usage. BLE devices are engineered to last for months using just simple coin-cell batteries. This technology is then used to create Bluetooth smart, which provides even more features from standard Bluetooth. These features are:

- Ultra low peak power consumption (BLE)
- Ability to run for years on small batteries (BLE)
- Lower implementation costs
- Multi vendor interoperability
- Enhanced range

Since BLE uses significant less amounts of energy than regular bluetooth, it has become popular in use for small items, which are desirable not to have to charge often. One example of this is heart rate monitors. Using BLE allows the monitor to be used for a long period of time without having to replace the battery. Other uses include smart watches and pedometers.

### 2.3.2 Game engine and Libraries

When creating video games, there are two paths a developer can take. The first is choose a programming language and start creating a game from nothing. This will require the use of some library to draw graphics, access inputs, handle screen coordinates and sometimes libraries to handles images internally within the program (translation, rotation, scaling). The first task is to then create a framework to build the game on. This framework will need classes to create interface objects, draw to the screen and handle all required events. Only when this is complete can the actual programming of the game begin. This framework is what is called a game engine.

The other option is to use an existing game engine, which has the entire framework already made. One of the earliest examples of a game engine includes the Pinball construction set (1983)<sup>8</sup>, Garry Kitchen's GameMaker<sup>9</sup> and Adventure Construction Set<sup>10</sup>. These early game engines were in reality more akin to what would be labled a library in modern development.

---

<sup>8</sup><http://archive.org/stream/ahoy-magazine-05/Ahoy.05.May.1984#page/n47/mode/2up>

<sup>9</sup>[http://www.garrykitchen.com/product\\_history/garry\\_kitchens\\_gamemaker.html](http://www.garrykitchen.com/product_history/garry_kitchens_gamemaker.html)

<sup>10</sup>[http://www.gb64.com/oldsite/gameofweek/7/gotw\\_adventureconstrset.htm](http://www.gb64.com/oldsite/gameofweek/7/gotw_adventureconstrset.htm)

It was only with the advent of 3D games in the 1990s. First Person Shooters like Doom and Quake were the very popular. The developer of the game, Id Software, created a core version of their games, with all the assets removed. They then sold this core to other developers to add their own assets and change the code as they see fit. Id software are still developing game engines, with the latest engine, Id Tech 5 released 3 years ago. The original quake engine has since been made open source under the GPL license and many new game engines have evolved from it, many of which can be seen in figure 2.3.

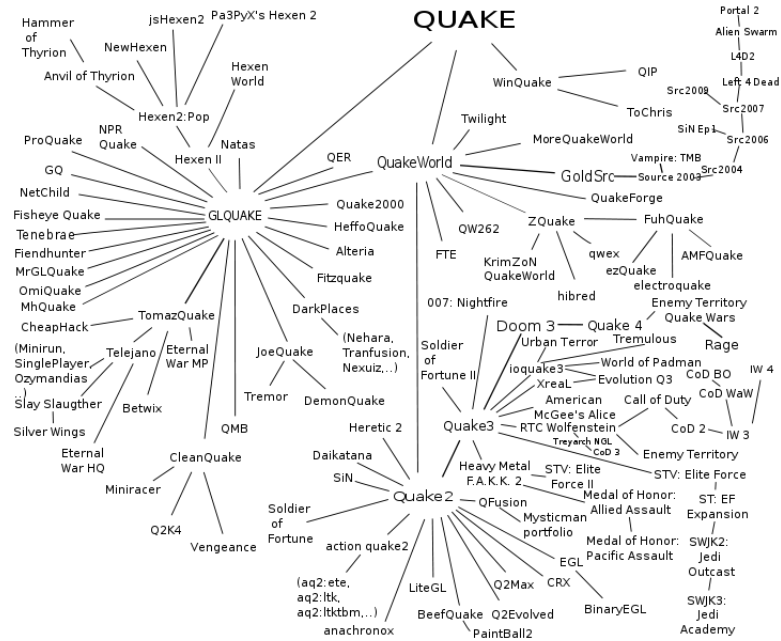


Figure 2.3: All derivative version of the original quake engine

Since game engines evolved from what would now be called libraries, the two terms are often used interchangeably with regards to creating games.

## 2.4 Summary

Game development is not new, and there has been plenty of work in this field already. This gives a rich set of theories and practices that can be used when creating a game. There has also been a significant change in the technology available to help create games in recent years, and there is also technology that has potential to be used in games but has not been utilized significantly yet. Bluetooth Low Energy and GPS are examples of this, and they will be used in this project to create a game that will be fun for children and adults alike.



## Chapter 3

# Conceptualizing the game

This chapter will use the concepts described in chapter 2 and try to create a concept using the theories and research that has been completed. It will show two of the most important parts to creating a game; create a high level document and a game design document. They are both used to describe a game, but they have different intended audiences.

### 3.1 Concepts

There are a few concepts that are integral to the project. These are ideas that are specific to the game and will be described below.

#### 3.1.1 Beacons

Beacons are specific points on a geographical map. There are two types of beacons. GPS and Bluetooth Low Energy (BLE). GPS beacons are used in places where BLE beacons would not be desirable. Outside, in fairly remote places are not the best place to have BLE beacons. They can be vandalised, stolen or run out of battery. It would be possible to use a GPS beacons as a backup, but then there is no point to having the BLE beacon there in the first place. GPS beacons also have a downside, they can't be used accurately indoors. In situations where the beacon is indoors, f.ex in a specific room in a building, a BLE beacon is used instead.

For BLE beacons, the location on the map uses a GPS coordinate to give a general sense of which building it is located in. There is also a clue written in text that gives a sense of where in the building the beacon is. The player can then look through the building and the game will notify the player when the beacon is within range. This is done to give a sense of discovery to the player, and make them feel like they are exploring without being directed directly to the beacon.

There are also other ways to implement the beacons. It would be possible to use NFC technology. This would require the players to physically touch a beacon with their phone. This would work for some scenarios, but is not used in this project because NFC is not available on as many devices as GPS and BLE. This is discussed more in section 5.4.1.

### 3.1.2 Guardians

Guardians are the character each user has in the application. There are not any ways to customize the application, but it would be desirable to enable customization were they application to be released on the market. Customization allows player to have more of a vested interest in their character and makes players take more of a interest in their character and their progress [13].

The guardians have a set amount of Stats; strength, dexterity and Intelligence. The Stats decide how the guardian fights in battle. A guardian with higher strength will do more damage and have more health, while a character with more dexterity will be able to evade damage more often and will also have a higher change of hitting the attacker. Wisdom affect the special attack and modifies how much damage they take from other special attacks.

Guardians also have a level modifier. Each time a guardian either defeats another guardian or picks up an experience beacon they gain experience. To advance to a new level a set amount of experience is required. Each time a guardian levels up, they gain more Stats and the amount of experience to gain the next level increases. This will give the player a sense of progression, by giving a compounding reward for winning battles.

## 3.2 Server

The backend server for the game should use a REST API. This decision was made to ensure maximum compatibility. http is one of the most cross compatible protocols that exist for transferring data over the internet. Http is supported by all desktop computers and mobile devices, smart TVs support http, and there are even toaster or coffee makers that support http to display rss feeds while bread is toasting or coffee is brewing. The amount of data is also sparse, as data is transferred as plain text. This is benefit when creating a mobile application due to the potential of limited bandwidth, especially if the player as at a beacon in a remote location. The full design is not described in the game design document, but will be described in chapter 4. There are alternatives to http; CORBA or SOAP are to examples. The alternatives are not very viable though, as the interfaces available are limited by the game engine that is used to create the game and at the time of writing there are no game engines that support either of these natively.



### 3.3 High level concept

The first step in creating a game is writing a high level concept. The high level concept is a brief description of the concepts in the game and what the ideas behind the game are. The high level concept is an *external* document. The document is often used to pitch the game idea to investors or game producers, with the hopes of creating interest in the game. It is often described as an elevator pitch, a brief description that can be given within the confines of an elevator ride.

The high level concept was created as a haiku deck. This allows for easy presentation of the concepts in the game and a simple overview of how the game is put together. The entire haiku deck can be viewed in appendix B.

The high level concept is a very brief overview of what the concept and ideas for the game. It is not meant to be documentation or give any technical insight. The high concept can also be used to help focus the game, by receiving input from potential players and other developers. If the high concept does not sound interesting to the target demographic, the game will need to change already at this stage. If the concept is not solid at this stage, it can create problems later in development.

### 3.4 The Game Design Document

Once the high level concept was created, some minor gameplay elements and mechanics were changed based on input received on the high concept from players and other game developers. The next step was to create the game itself. The game is designed in Unity3D. The reason this technology was chosen will be give in more detail in section 4.1.

When creating a video game, it is very common to create a design document. The design document is an internal document that is only shared with designer, producers, programmers and anyone involved with the creation of the game. The design document should give a detailed description of what the game is, what mechanics it uses, what the visual design of the game should be and any other relevant information to creating the game. It should also describe the Story bible. The document should be detailed enough that the game can be recreated by reading the document and creating the contents of it. The story bible is an integral part of a game. For some simple puzzle games it is not as important, but any game that has any characters or any goals for the player should have a story bible. It gives the player a reason to do a desired action.

When creating the game design document, which can be read in appendix A, which was created after the high level concept and research was completed.

### 3.4.1 Gameplay

#### Risk/Reward

The main risk reward element is the battles. When two characters battle for control of a beacon, the player has a choice of three different attacks. The attacks use Rock-Paper-Scissors mechanics. This is a very simple design, but one that is used often even within modern games. It emphasises the reward more than the risk, since the player has no idea what the opponent will choose and can't change the amount of risk, each attacks has an equal amount of risk. The reward is a compounding reward. The team the player belongs receives a global score for their team, and their guardian receives experience and can level up. Loosing a fight can be seen as a compounding or ultimate loss. If the player looses their last beacon, it set's their score to zero and they are starting from the same state they started the game (which is equivalent to starting a new game).

Another risk element is healing the guardian. Since the guardian can not heal while deposited at a beacon, the player can take a risk of moving their guardian to their phone to heal. This means that a beacon they controlled can now be taken over by a player from the opposing team. This is a risk that the player has more control over. They decide if the health is low enough that the risk is greater to keep the guardian at the beacons, than depositing it on their phone to heal. Here the reward is not as tangible for a player, they don't get any points or score for healing a guardian. It can be seen as a compounding reward. Loosing a beacon because the guardian was deposited on the users phone, can be seen as mixture of a compound loss and an ultimate loss. The game is not over for the player, but significant progress can be lost.

#### Difficulty/challenge

#### Bartle Test

Since the game is a multiplayer game, it is possible to use the bartle test to help target the game to different types of players:

##### Achievers

The point system that the game implements will give achiever a goal to reach for. Making their team own as many beacons as possible will be the goal for achievers.

##### Explorers

This is integral to the design of the game. Since the game also takes place in the physical world, players are encouraged to explore their real life surroundings. This can be very rewarding for explorer type players. This can be improved further by hiding beacons in remote locations, or creating special beacons that are only visible if you are within a certain distance of it, making it invisible to players who are not close enough.

##### Socializers

This group of players want to socialise with other players of the same game. Targeting this type of player is easiest by implementing a chat

system. This will not be a priority, and is categorized as a "Would be nice" feature.

### **Killers**

The battle system in the game will give Killers an opportunity to fight against other players and will appeal to this category of players.

### **Mindset, Art style**

Another significant part of creating a game design document is creating the setting and mindset for the game. This will help infer what the setting is and what the game is trying to tell the players. This can be done through artwork, music, sound effects, animations and more. Sometimes this uses genres which are already popular and build on that. Examples of this are Steampunk, Cyberpunk, and Robotics. It is also possible to create new genres and settings, which will then require more explanation to ensure all involved parties can agree on what the mindset and feel of the game is.

This project uses it's own simple mindset, where the player has to help save the world with the help of alien robots. The game should have a simple comic feel to the art, with very earthy tones. There should be two distinct worlds, the world around the player (the physical world) and the world where the guardians live (the augmented reality world).

## **3.5 Alternatives**

There are alternative ways to implement the gameplay. One way would be to no use the rock paper scissors mechanic for battles. An alternative way is to give each guardian a different types of attacks that have different effects. They could for example have one attack that lowers the defence of the enemy, making them take more damage from a regular attack during the next round. This creates even more of a risk, in that the player can choose not to do damage for one round, with the reward of doing more damage the next round. This was not chosen because a rock-paper-scissors style battle is simpler to implement, while also giving the player a feel of risk.

Adjusting the amount of rounds required to win for a battle to be won increases the effort required to win a battle. Looking at the equation in chapter 2

$$C + E = \textit{Reward}$$

it shows that increasing the effort will then also increase the reward the player feels. Because of this, the amount of rounds required to win a battle will increase as the level of the players guardian increases.

The battle graph does not have to look at the big picture, it can make differences on the small scale too.

## 3.6 Summary

Using the information and theories researched in chapter 2, it is possible to create a game that has the potential to be both fun and rewarding. A big part of creating a game is planning the development. Creating a high level design and receiving early feedback is essential to creating a game that players will enjoy. Using the feedback and using it to create the design document it is now possible to proceed with the actual creation of the game.

## Chapter 4

# Implementation

This chapter will show how the design document was implemented and how the mechanics were made into a fully fledged game. Choosing the right technology for a project can make a big difference, and choosing the wrong one can both hinder the project and all together slow it down. After narrowing down the choice from many different libraries and game engines, the choice came down to three different options. The simplest was to develop the game using the native SDKs for Android and iPhone. The main problem doing that is that the iPhone SDK can only be developed in the OSX operating system, which was not available for use continuously through the project. The author has never written software for IOS devices either, so the programming language (objective C), the framework(IOS SDK) and the IDE (Xcode) would have to be unfamiliar at the start of the project.

When comparing the different available technologies, there were several criteria that were considered. The most basic requirement is support for 2D graphics and 2D sprites. It also needs to have support for development of mobile games. This creates a baseline for which libraries/game engines to look at. The three that were chosen was Unity3D, which has gained massive popularity and is currently estimated to have a 45% marketshare<sup>1</sup>. Another option is libGDX, which is a very popular library that is used for many mobile games. It is open source and has a big community around it. The final option is using the native SDKs for iOS and Android.

The main issue with libGDX is that it does not allow the use of accelerometers and it does not support GPS or BLE<sup>2</sup>. Since the project is open source it would be possible to write plugins or modify the source code to allow it access to both GPS and Bluetooth, but that would be a full project in it's own right.

Table 4.1: Comparison of different libraries/engines to create games with

| Technology                           | Unity3D | libGDX | Native                   |
|--------------------------------------|---------|--------|--------------------------|
| Support for 2D Graphics              | X       | X      | X                        |
| Mobile device support                | X       | X      | X                        |
| Out of the box support for GPS       |         |        | X                        |
| Out-of-the-box support for BLE       |         |        | X                        |
| External library support for GPS/ble | X       |        | X                        |
| Cross platform                       | X       | X      | X                        |
| Familiarity                          | X       |        | Not for game development |

## 4.1 Game engine

Looking at table 4.1, libGDX does not fill many of the criteria boxes. What is especially important is that it does not have support for GPS and/or BLE. This is such an important part of the game design that libGDX can easily be dismissed. The choice is then between using the native SDKs or using Unity3D. While using the native SDK does tick all the boxes, it does have two major negatives. The biggest problem is that it would require the game to be made twice if it is going to be released for both android and iOS. Also, having no prior knowledge of iOS development, it would probably take the entire time of the project just to learn the tools and frameworks for creating a game for iOS. Since the project is only over a very short timespan, familiarity with the tools can make a big difference. Therefore the decision was made to create the game using the Unity3D engine.

Unity3D is an increasingly popular game engine and IDE. It has just been released in a new version, Unity3D 5. Unity supports both 3D and 2D games, and mobile platforms. It can compile games for many platforms (Windows, Android, iOS, Playstation3/4, Xbox, and more). It is available to use for free, but for mobile applications it will have a splash screen with a unity3D logo when the client is started. In the event that the game is in a state to be published, it is possible to pay for the Unity3D plugin for mobile development which will remove the splash screen.

## 4.2 Middleware/Backend server

The server is a REST server, using only JSON to encode messages. The server is implemented in python using the python-flask framework. It is based on a MVC architecture and used CRUD principles. python-flask has a class that is created

<sup>1</sup><https://unity3d.com/public-relations>

<sup>2</sup><http://libgdx.badlogicgames.com/features.html>

to help make Method views, which allows a class to be defined with its own GET, POST, PUT and DELETE methods. This allows the server to be very modular. If a new models needs to be added, the model is first created (using flask-sqlalchemy) and then the View is created by the MethodView classes. There is no designated controller for the server and the MethodView classes also take care of the control of the data.

There is python module, flask-restful, that was implemented for a short amount of time. The module was out of date and exception handling the flask framework had changed since its last update. This allowed many silent exception to pass, which made debugging the server extremely difficult. The module was then scrapped and the more stable MethodView option was used.

One module from the flask-restful remained in use though. The "reqparse" module allows parsing of data that is sent as part of the request body. It has the benefit that it looks in both the JSON body and plaintext body of a http message. It will always return the request body as python dictionary object. This allows the server to handle the request no matter how it is sent to the server as long the the body data is JSON compatible.

The REST API follows the design pattern from the book "Rest API design rulebook" [16]. It does change the URI format from the defined:

```
uri = scheme "://" authority "/" path ["?" query] ["\#"
    fragment]
```

to

```
uri = scheme "://" authority "/" path "/" resource "/" identifier
```

The query is now seen as part of the request method. This is done because it creates a cleaner API that takes advantage of the different request methods. The API created for the server also only allows CRUD functions. If the server was to implement more complex queries, f.ex retrieving all the users with a certain name or search through the guardians for a certain number of locations, the URI scheme would be changed to use the URI scheme described initially above, by M.Messe.

For each resource and endpoint is created. The endpoints can be changed in the server configuration file. The default is:

```
"/api/[resource name]/"
```

so for example issuing a PUT request to the url

```
"/api/beacon/5"
```

will update the beacon with the primary key 5. The new data is provided in the body of the message, formatted as JSON (often called the JSON body in flask documentation).

Table 4.2: Middleware comparison table

| Middleware                        | Parse.com                                 | Python-flask server                                      |
|-----------------------------------|---|--|
| Maintenance                       | Server need to be maintained and serviced | handled by 3rd party                                     |
| Support for Coordinate data types | built in (GeoPoint)                       | would require additional work                            |
| Hardware and scalability          | ”Cloud”                                   | Requires server hardware, and will require more to scale |
| Libraries and API                 | support for javascript, REST and Unity3D  | Rest Only  |
| Security                          | no control and no control of the data     | full control   |

### 4.2.1 Parse.com

The backend server was dropped in favour of using parse.com (Parse). Parse has many benefits to using an own server. Table 4.2 shows a short comparison.

Parse is a distributed key value storage, specialising in the mobile market. It has built in support for Android, iOS and Unity3D. The most useful feature of parse in regards to this project, is it’s built-in data type for GPS coordinates. This is build into the query system. It is possible to query if an object is close to a given coordinate. One example:

```
player = Parse.query.(withId=1)
beaconsClose = GeoPoint.query.WhereNear(player.position)
```

This is very valuable when the game is built on using GPS coordinates as a major part of it’s gameplay. Creating this from scratch on a self built middleware server would take considerable more effort than using the functions already available through parse.

Table 4.2 shows a comparison of the server initially developed and using parse. The table shows that parse has benefits with regards to server maintenance, hardware and scalability. Since parse.com is a cloud solution, the scaling is done by the cloud provider, and is not something that is required to be monitored by the application. The hardware and maintenance of it is also provided by the cloud provider. Parse also has the benefit of it’s own data type, which allows coordinates to be used with ease. This is a very nice feature to have when the whole game is based on GPS coordinates and is the core part of the gameplay. Parse also has a Unity3D plugin, which allows it to integrate seamlessly with Unity3D, making it ideal for games.

The only downside to using parse, is that there is no control of the data. Since the data is stored on a cloud provider, the application will stop working if



something happens to the provider, and nothing can be done to fix it, other than wait for the provider to fix it themselves. There is also the issue of what happens if parse suddenly disappears, due to financial or similar issues. There is a risk the data would disappear in such a situation, making the game unplayable and potentially resetting all user data. This would be a very bad situation, and if the game was to be rolled out on a big scale, it would be important to enable some backup to save the state of the game regularly.

There is also the issue of security. There is no way of knowing exactly how secure the data is. Since the parse system is closed source, there is no way to make sure that it is safe from intrusion. For this reason, no sensitive data will be stored in the parse key-value store, and all passwords will be cryptographically hashed to ensure that the passwords are safe.

## 4.2.2 Other possible server options

There were not many other server systems that were considered. Since parse was only discovered almost 2 months into the development process, time was precious commodity. There are other big scale server solution that could have been looked into, like Amazon web services, which could have used Amazon DynamoDB, which is a simple highly available key-value store. But even after briefly looking at other solutions, none had an equivalent data type to the geopoint data type that Parse has. The geopoint type is really what made parse really useful, as without that, Parse is just a simple key-value store like any other, and any key-value store could have been used.

## 4.3 The game client

The game client is written in Unity3D. It is made for Android and iOS devices.

A singleton class is used to handle all game state variables. It is created when the application starts, and is transferred from scene to scene. C#<sup>3</sup> and unity don't have any language specific ways of implementing a singleton, a workaround is used. When the application starts, it creates a new singleton. When it is created it checks to see if it's private class member "instance" is defined. If it's not defined, it defines the instance member to be itself. The code for the workaround is described below:

```
public class GameController : MonoBehaviour
{
    public static GameController Control;

    void Awake(){
        if (Control == null) {
            DontDestroyOnLoad(gameObject);
            Control = this;
        }
    }
}
```

---

<sup>3</sup><https://msdn.microsoft.com/en-us/library/ff650316.aspx>

```

    }else if (Control != this){
        Destroy(gameObject);
    }
}
}
}

```

### 4.3.1 Saving and loading data

Unity3D provides a class that is made for simple storing of data; PlayerPrefs. This class works the same way the Shared preferences work in the Android SDK. It stores data as key value pairs in plain text in a predesignated place in local storage. On windows it's stored in the registry under

HKCU\Software\[company name]\\$[product name] key, where company and product names are the names set up in Unity3D project Settings.

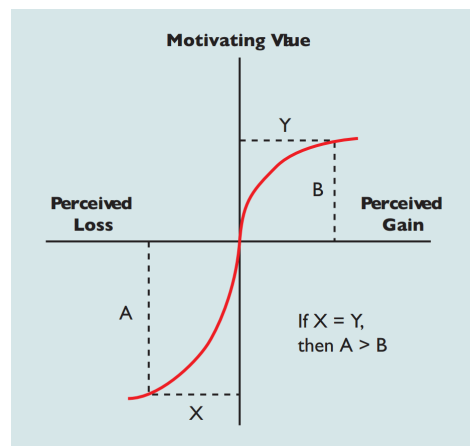
On android it's under

```
"/data/data/[Appname]/shared_prefs/appname.xml"
```

and on iOS devices it's at

```
"/Apps/[Appname]/Library/Preferences/[appname].plist"
```

The information in this data store is saved in plain text, which means that they can easily be edited by a user if they have a little knowledge of how unity applications store player preferences. The class only support certain data types, as everything is saved as a serialized string, and does not support saving binary data.



For this reason, an own data saving class was created for the game. It uses a BinaryFormatter class to store the data into a binary file. While this is not impossible to read using the correct tools, doing so on a mobile device takes considerable more effort than editing a plain text file. This can be seen as a type of "security by obscurity".

Figure 4.1: Losses carry more value[...]

In a paper by Ryan west [22] it's demonstrated that if making the effort required higher than the perceived gains of doing an action will make it less desirable to do. The binary file created by the client application is by no means safe, if compared to Kerckhoffs's principle [12]. While not 100% secure, the effort required to do so on a mobile device is believed to outweigh the gains to be secure enough for it's own application. Figure 4.1 shows that losses carry more value compared to gains when both are perceived as equal. For non-zero values, if value of loss (X) = value of gain (Y), then motivation of loss (A) > motivation of gain (B)[22].

Another reason for using the binary formatter to save and load data is that it allows the storage of binary data. The robohash avatar each user receives when they sign up to the game is saved locally on the device. This allows it to be displayed at times when there is not internet connectivity on the device. It also saves on bandwidth when displaying the avatar and reduces load times when it has to be displayed as the save file is loaded into memory when the application starts, so the image does not have to be read from local storage every time it has to be displayed. The GameState singleton class handles all the loading and saving of data.

## 4.4 Summary

The game is built using Unity3D, which has the benefit of being compatible with all mobile devices on the market. It is also a tool that is familiar, which can speed up development significantly. This does have some downsides. Unity3D does not have access to low level devices on smart phones and have to rely on 3rd party plugins to enable access. The functionality available is then at the mercy of the plugin writer, and there are no guarantees that they work well, and the functionality can be difficult to customize.

There was considerable effort put into creating a backend service for the game. This was later abandoned for Parse.com. Parse allows the use of a coordinate aware data-type, which is very useful when the game is based on the concept of fixed points on a map. Creating this for the initial server implementation would take considerable effort. For this reason, parse was selected to be used as the backend for the game.



## Chapter 5

# Reviewing the work

Once the game was completed, it was possible to see if the goal of creating an exciting and engaging game. This chapter will review the final result and see how the game actually ended up and how much it has (or has not) changed since the initial design. The chapter will also look at the decisions taken during the development, how the game is tested and what can be done in the future to improve the game.

### 5.1 The final game

The final version of the game does not have all the elements set forth by the game specification. It allows players to register when starting the game and have their own avatar. It does however not allow the player to choose a class. The battle system would have to be revisited to allow for different sets of skills to give advantages and disadvantages. The game does allow players to find beacons and fight for them and take ownership of them. They also do not heal while deposited at a beacon, and heals while "stored" on the players mobile device. The game does not tally any points for the two teams at the moment, but the information is readily available in the Parse key-value store, and it would only take a few lines of code to display how many beacons each team owns.

### 5.2 Scenes

Below is a description of all the scenes and menus that are in the completed game. It describes their finished state and what could be done to improve them.

### 5.2.1 The main Menu

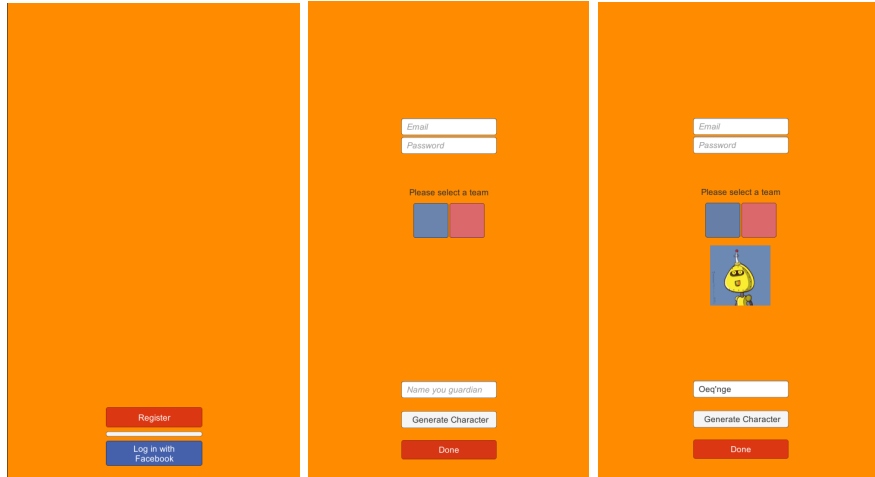


Figure 5.1: The registration screen

The main menu only shows if the user has not registered their application. The main menu consists of a logo and one button to register the application. When the user presses the register button, they are prompted to write their email and a password. This is stored on the parse server as a new user entity. They are then prompted to select a team, red or blue. There is also a button which generates a random character for the player. An avatar for their guardian is then created using robohash.com.

Robohash is a web service that will accept a string, create a hash of it and generate a robot based on the hash created. One big benefit of this is that it is not required to always store the players avatar. As long as the name of the avatar is known, it can be sent to the robohash service and the same robot will always be returned. The look of the guardian is only cosmetic, it does not make any difference for the gameplay.

The main menu does have animations. When first presented with the main menu, there are only two buttons. When one of them is pressed, they are moved outside the viewable area. All animation is done using Tweens from the iTween unity plugin. Tweens were used because they are simple to use and they create a very smooth and fluid animation. It also allows for effects where the animation eases into places, or bounces a little when it hits its target. The common alternative is to animate the buttons using vectors and updating the position of the button for each frame update. While this is also a good solution, it takes more lines of code and the result is not as fluid.

Once the user has filled in all the required information, they can press the "done" button and all the information is saved, first to parse.com, and then it is also added to the game control object, which will enable the information to be used

to other scenes and also allows saving of the information to disk.

The registration screen is created to be as simple as possible. Since the game is targeting 10-14 year old children, there are limitations on the data that can be retained about users<sup>1</sup>. For example, the rules for the "Children's Online Privacy Protection Act" (COPPA) which is a US Rule from the Federal Trade Commission states that any personal identifiable information of children under the age of 13 can not be gathered without consent of a parent. Emails and usernames are not protected under COPPA, so that is the only information the player has to enter to play the game.

### Creating a random character name

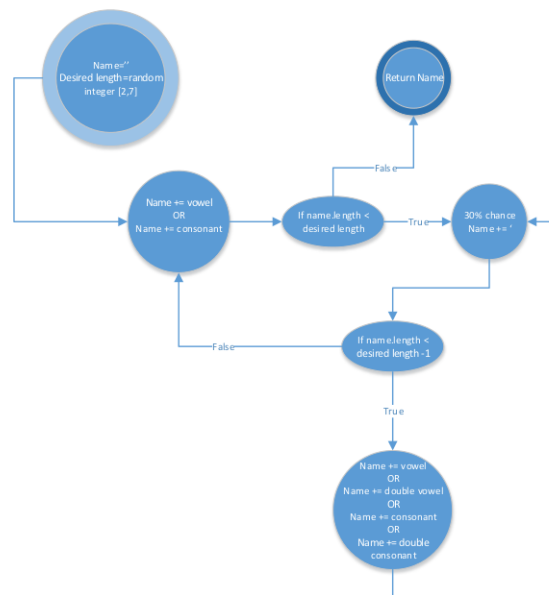


Figure 5.2: Program flow for the character name creator

name. It starts by choosing a random length for the name. It then chooses the first character in the name, which can be either a vowel or a consonant. It will then loop over the length of the name, adding wither a single or double character, which is chosen based on which character was chosen earlier.

There are 27 letters in the alphabet, and " ' " is added as a valid character, so there are a total of 28 possible characters. There are also 5 different possible name lengths. If a " ' ", it will not count towards the length of the name, so

<sup>1</sup>[https://www.ftc.gov/tips-advice/business-center/guidance/complying-coppa-frequently-asked-questions#General Questions](https://www.ftc.gov/tips-advice/business-center/guidance/complying-coppa-frequently-asked-questions#General%20Questions)

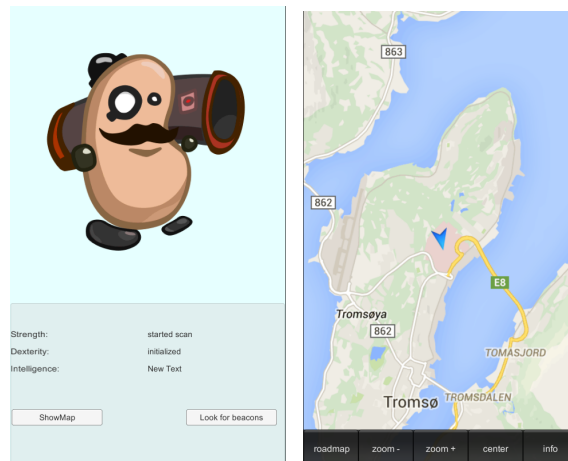
The most important part of creating a random character is generating a completely random name. Using some basic knowledge of linguistics, we can define a name following a certain pattern. Four basic choices are created, single and double character vowels. After a vowel (single or double) there has to be a consonant(single or double). A name can only start with a single character (no double vowels or consonants). It also allows a " ' " to appear between characters. Figure 5.2 shows the programming flow when creating a new character

the name can be even longer if several of them are added. This means that the probability of of a name appearing twice is very low.

The limiting factor in preventing two identical characters is not the name, but the hash used to create the robot. There is a bigger chance that two names will create the same robot than there is of the same name appearing twice.

The names that are produces are intentionally other-worldly, sounding alien to most languages. There is also a high density of vowels, making the names have a sound that is close to what a robot could make with a simple synthesizer. This could also have potential to make sound effects that have sounds very close to the names of robots. Examples of names are Oota, Aloora and Eertaan.

## 5.2.2 Guardian scene



(a) Early version of the character screen

(b) Early version of the map scene

Figure 5.3: The main screen

This scene is the main view for the player when they have registered. Here players can see their avatar and the statistics and abilities of their guardian. They can also press a button to see if there are any beacons within reach and take control over it or fight the guardian currently protecting it. Figures 5.3a and 5.3b show the final design for the scenes.

They can also press a button to view a map. This map will display the players current position on the map and any beacons. If the beacons has a guardian protecting it they can press the beacon icon to see the guardian protecting it and the stats of that guardian. That way the play can judge if their own guardian has the stats to beat the enemy guardian. If there is no one protecting it it will be displayed as a black beacon and the user can choose to protect it if they desire. The map uses google static maps, so the design of the actual map can not be changed much and the interface for the map screen is embedded with



the GPS Unity3D GPS plugin. While it can be changed, it was not prioritised for the project and the map interface should be considered future work

The game uses information from the map to create a dynamic image for each guardian. It checks the colour of the position on the map and assumes that green means forest and blue means water. It also checks the elevation information for each guardian and checks the time of day. It then uses all this information to choose from the set of available backgrounds shown in figure 5.4.

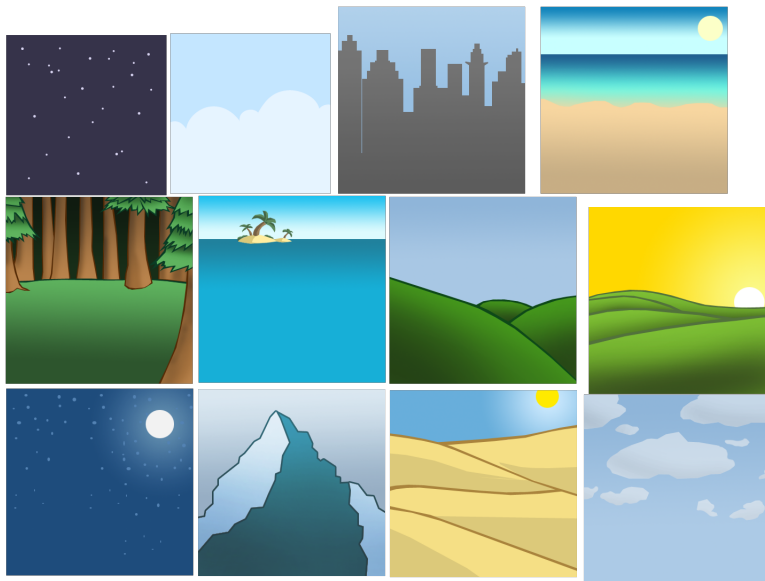


Figure 5.4: The available backgrounds for the guardians placed on the map

### 5.2.3 Battle scene

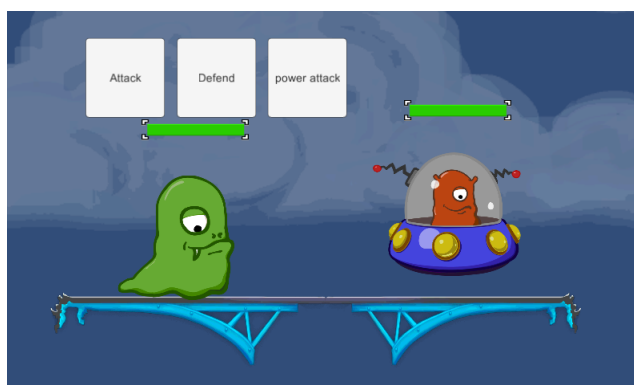


Figure 5.5: Early version of the battle scene

The battle scene is displayed if the user decides to fight for control of a beacon. The player can choose an attack type (attack, defend or charge) and an animation will play and show if the attack was successful or not. Each guardian also has a health bar. This gives the player feedback on how much health their guardian has. The scene has idle animations, attack animations, defend animations and evade animations. Figure 5.5 show the final design for the battle scene.

The battle scene is made to be as simple as possible. Since the game is aimed at children, the controls should be as simple and intuitive as possible. There are only three choices, and the player receives instant feedback if the attack was a success or not. The animations and art styles are also made to be quite cartoon like. The art in figure 5.5 is place holder art, but if the story of the game was about aliens, not robots, it would be acceptable art for the game.

### 5.3 Testing

The first test of the game tested with three GPS beacons and two BLE beacons. The beacons were placed at the UiT, The Arctic university of Norway campus, and the GPS beacons were placed at random locations in the woods close to the campus. By editing the parse.com tables two of the beacons were selected to be owned by team red. An account was created, with an avatar on team blue.

The GPS beacons were easy to find and allowed easy discovery. The BLE beacons were a little more difficult to find. The BLE plugin for unity is slower than anticipated and can make finding beacons difficult if looking for them while walking. No attempt was made trying to improve the performance of the BLE beacons and this is considered a potential for future work.

There was also some player testing. Five Beacons were placed within short walking distance of the university campus, and two BLE beacons within building located around campus. The beacons were places similarly to the initial test, but more spread out. This was tested with four players, two on each team.

The feedback from the user testing was mostly positive. The BLE beacons were hard to find at first, as the only clue they had was which building the beacon was and a description of where it was located. This could be improved by making a better textual clue to the player. The feedback in the gameplay was that it was fun, but very simple. This is not necessarily bad feedback, as the testers were above 20 years of age, while the target group is much younger. The game also didn't see quite as deep as the players would like. This is something that is intended to be fixed with future work.

It would be favourable to do actual user testing with children within the target demographic. This was not possible to due to time constraints. One simple way of doing this would be to ask schools, and use a period of school and enter a classroom and use the class to test the game. This has been done with success with other games [15]. In a play testing scenario like, it would be important to limit both the number of players and the number of beacons. There should be

as little interference with the game while it is under testing. This allows the player to discover the game for themselves, and the game aims to be intuitive enough that children can use the game without requiring any instructions.

There should also have been some player testing on beacons placed in remote locations. This would require play testing over time, preferably more than two weeks. Placing beacons in remote places require more effort for the player to physically move to the beacon. This is something that might actually look like a very good idea, but actually be not as fun when set into practice. It is therefore very unfortunate that this was not tested in time for the project to finish.

The parse key-value store was also tested with 1000 beacons and 1000 guardians. The tests showed no performance loss due to the scale, on neither writes or reads. Since parse.com uses a proprietary closed source solution, it is hard to know how different workloads will affect performance, unless significant amounts of testing is done.

## 5.4 What could be done differently

It took a lot of time getting the technology together and making all the pieces work together. A decision should have been made earlier on how to create the beacons, and how to interact with them. Some more research should also have been done into the plugins available for unity3D. Both the beacons plugin and the GPS plugins are not very extensible. It would perhaps have been better to write plugins to access BLE and GPS functions using the android NDK. This might have taken a little more time to implement, but it would make it easier to extend at a later date.

There should also have been more planning for some kind of web frontend for the game. This was an afterthought and there was no planning for it. This means that there is nothing to compare the end result to. It should have been just as big a priority as the game client itself. This shows that a game specification should not always just be about the game itself, but also any other tools around the game. This allows any tools to have a proper priority within the game ecosystem and not be an afterthought. The administration frontend could have used the same web server and site as the official site for the game, reducing the time required to develop. Since this was not finished, it is also something that could be done in future work.

### 5.4.1 Alternative technologies

There were some very important decisions made when it came to choosing the technology to use to create the game. In hindsight, it might have been a better solution to create the game using native mobile SDKs. Unity3D made it very quick to create the game, but made it very hard to use low level devices like GPS and BLE. So while it was easy to create the game itself, it was much harder to use both GPS and BLE. They both had to rely on Unity3D plugins, and the

quality of unity3D plugins can vary. The GPS plugin was fairly easy to use, but it was much harder than expected to implement and use BLE in Unity3D. The amount of BLE functions was also limited to the functions the plugin provides, as the BLE library was provided as a compiled library (dll file).

There are also other technologies that could be used, but were not available as Unity3D plugins. One such technology is LTE-Direct [2]. LTE-direct allows device-to-device discovery. With this technology it would be possible to allow players to fight each other without having to go to a beacon to battle. It would also make it much easier to find beacons inside buildings, as LTE-direct is direction aware, and the game could point the player directly to the beacons position. LTE-direct is not currently supported in Unity3D. It also not as widely available as GPS and BLE as LTE-Direct is not officially released to the public yet.

Another option would be to use NFC technology. The reason this was not chosen is because NFC is not available on as many devices as BLE. While BLE is available on any device that supports Bluetooth v2.1 + EDR, which is a standard that has been available since 2007 and is available on most modern mobile handsets. NFC technology is not available on any iOS devices, which could exclude as much as 40% of the phone market<sup>2</sup> from playing the game. It is also true that any phone that supports NFC will also support Bluetooth Low Energy, so the it is considered a better solution to settle on a technology that is supported by as many devices as possible.

## 5.5 Future work

The first task for future work is to make it easier to commercialise the game. The original idea was to create a game that could be customized and have several instances of the game running at any one time and allow easy customization of the game assets. This was not completed by the end of the project. Since the game relies on the Parse key-value storage, it would not take much effort to implement the running of several instances. It would be simple to create an instance and give the user a pin code they can enter when registering for the game and the player would be connected to one distinct instance of the game. The main item missing from this is creating a frontend for administrating the beacons. A simple frontend has been created, but it does not support multiple instance of the game running at once in it's current state. It would be possible to have this instance connect to a distinct game using the same pin that the user uses. The frontend would require some sort of access control.

Allowing custom assets would require a change of how the game loads the asset shown to the player. This work was not prioritised because there was not an abundance of assets available. In this manner, it would also be a very important to improve on the assets for the game. Professional graphics and sound assets should be a priority. Music would help improve the game.

---

<sup>2</sup><http://www.forbes.com/sites/chuckjones/2014/06/04/apples-u-s-iphone-market-share-holding-steady/>

The story bible for the game is not expressed in any meaningful way. Therefore it would be important to express the ideas and stories that are the motives of the game more to the player. This will help make it easier for the player to get involved in the game and create more enthusiasm to play the game. If the player knows more about the reason they are required to do an action, they are more likely to do the action.

The game could be expanded on, but this would require significant planning. The game is very simple, but it is important to remember that this is by design. The game is meant to be simple so it is easier for younger children to grasp the game. That does not mean that the gameplay can not be improved on. On gameplay change that can improve the game is adding character classes that each have a different play style. These would be archetypes: Rogues/Theives, warriors/knight, Wizards/Mages. These are very easy for children to identify with and they each have a distinct personality. Knight/Warriors are benevolent characters that fight for good. Mages/Wizards are bookish characters that are not strong physically, but have great intellect. Rogues/Thieves are not benevolent or book smart, they are "street smart" and are more social characters. These are based on archetypes in Jungian psychology [10]. Using archetypes make it much easier for children to identify with the characters and can increase the emotional attachment to their own character. Clifford Mayes has written several papers on the subject of using archetypes in education for children, by making teachers follow Jungian archetypes [17, 18]



## Chapter 6

# Conclusion

The goal of the project was to create a game from scratch, that augments the world around the player. It is supposed to be targeted at children with the explicit goal of making them more active. It is hard to prove that this was reached without testing over several months. But the game is playable, is fun, and rewards the player for exploring their surroundings. We can compare this to the statements created in chapter 1:

- Create an interactive game that is aimed at children aged 10-14
- Create a game that forces the player to move outside to play. It should not be possible to just play from the couch

Early player testing shows that the game is both fun and engaging. Unfortunately there was not enough time to test within the target demographic. But feedback from other game developers has been positive and there has been plenty of good feedback for the high concept. Especially the idea of using the local community, like city councils or schools has received plenty of positive feedback

The game mainly used simple gameplay to help entice younger players. Art also plays a major part in attracting children, with simple graphics that are also very recognizable. The game especially succeeds in completing the second statement. It is not possible to just play the game from a couch. The player has to be active to play the game. Even if they place a guardian at a beacon, they will have to return to pick their guardian up to make sure it heals, forcing the player to revisit the beacon.

The game does not have its final assets in place, so the art design is a little muddled and not very concise. But the assets that are used are adequate for its purpose. The art is very cartoon like, which is the type of art that has an easier time appealing to children. It also makes it easier for parent to let the kids play the game, if they see friendly cartoons, compared to realistic robots, or scary aliens.

Since the game is made for mobile devices, it does require children to have access to a smartphone or tablet. But it is also possible to play the game as a family activity. A family could have on character they share on a device, and they can visit their beacon as part of a family trip out into the woods.

The game also succeeds in creating an Easter egg hunt type game, that can have several uses. Since the game is able to locate beacons inside building, it can be used as a way to explore a new area, like a university campus while at the same time letting the user play a game. All beacons have a name, which can be used to orientate the user as to what is significant about the particular beacons placement.

Planning is very important for game development. There are several concerns in game development that are not a concern in traditional software development. Arts, music, sound, story, gameplay are all elements that combine to create great games. This is on top of regular software development. This is also where the game can be developed more. Professional graphics, music and sound effects should be a priority for further development. Chat functionality would also increase the appeal of the game, allowing teams to plan and communicate what their plans are. They can agree when to withdraw guardians from beacons and make sure someone from their team takes it again and protects it.

The end product of the project is a game that is fun for children, and if they desire to play the game, they have to get up from the couch, go outside and fight other players. For this reason, the project is considered a success. There is still work that to be done, but the core experience is exactly what it should be; engaging and interesting.



## Appendix A

# Game Design Document

# Earth Guardians

## *Game Design Document*

---

Copyright Jørn Lomax 2014-2015

# Index

---

1. [Index](#)
2. [Game Design](#)
  - 2.1. [Summary](#)
  - 2.2. [Gameplay](#)
  - 2.3. [Challenges](#)
  - 2.4. [Mindset](#)
3. [Story](#)
4. [Technical](#)
  - 4.1. [Screen Controls](#)
  - 4.2. [Technologies and libraries](#)
5. [Level Design](#)
  - 5.1. [Themes](#)
  - 5.2. [Game Flow](#)
  - 5.3. [Scenes](#)
6. [Development](#)
  - 6.1. [Base classes](#)
  - 6.2. [Priority Schedule](#)
  - 6.3. [Must haves](#)
  - 6.4. [Should haves](#)
  - 6.5. [Would be nice to have](#)
7. [Graphics](#)
  - 7.1. [Style Attributes](#)
  - 7.2. [Graphics Needed](#)
8. [Sounds/Music](#)
  - 8.1. [Sounds](#)
  - 8.2. [Music](#)
9. [Business plan](#)

# *Game Design*

---

## **Summary**

The earth is very polluted, something has to be done about it. It is up to the citizens of the earth to help teams of robots clean up. But the robots have different ways of cleaning up all the pollution. Who will you help?

### **Inspiration**

GeoCaching, Ingress

## **Gameplay**

Battles use a paper-rock-scissors systems. Players can choose whether to attack, defend or charge.

Charge -> wins against defend, loses against attack

Defend -> wins against attack, loses against charge

Attack -> wins against charge, loses against defend

## **Challenges**

The challenges are mainly in the the physical world. The players have to travel to a certain spot on a map to challenge other players or take beacons. There is some challenge in fighting other players, but battles are still subject to randomness

## **Mindset**

The player has to help save the world. There are two teams trying to do this, each with their own way of doing it. Both have pros and cons and the methods should be balanced. Guardians from each team can not move in a polluted world, the player has to move them safely in a capsule, their phone

## **Game world**

There are two distinct worlds. There is the game world, which is happening on servers

and consists of bits of data. There is also the physical world, where players have to move around in. The game is aiming to *augment* the physical world

## *Story*

---

The earth had been damaged beyond repair from pollution and greenhouse gasses. There are two races of alien robots that want to clean up. These robots are damaged by pollution, and can not move around on earth due to the severe amount of pollution

The two races of robots have different ideas of how to solve the issue. One team wants to recycle rubbish to create new items. The other race wants to convert the rubbish and waste into electricity, reducing the earths reliance on fossil fuels. They both want complete control of all the rubbish and waste that is on the ground.

The robots had to move from their previous planet due to pollution, and now they want to save earth from suffering the same fate, making it impossible for humans to move outside due to pollution

## *Technical*

---

### **Screen Controls**

The game is designed for mobile devices. All controls will be point and click on a touch screen

### **Technologies and libraries**

The game will be created in unity3D, using c# scripts and the Unity3D "2D" mode. It will only use the free version of the program

The game will require the use of GPS and BLE beacons. This will require libraries from the unity asset store.

## Server

The game requires a backend server. The server has to handle user registration, beacons and their locations, and all the guardians the players have. The server should be built using a rest API to allow access from multiple types of devices and allows a high degree of compatibility. The server design will not be outlined in this document.

# Level Design

---

*(Note : These sections can safely be skipped if they're not relevant, or you'd rather go about it another way. For most games, at least one of them should be useful. But I'll understand if you don't want to use them. It'll only hurt my feelings a little bit.)*

## Themes

Protecting the earth. The earth is under attack due to the amount of pollution in the air and litter on the ground

## Game Flow

1. Player has to register an account
2. Player has to choose a team, a name and an avatar
3. Players can view their character and it's progress
4. Players can search for beacons on a map
5. Players can take an non-owned beacon and take it for themselves, or they can fight another player for a beacon if it is already owned

## Scenes

Register page:

# Register Page

Name

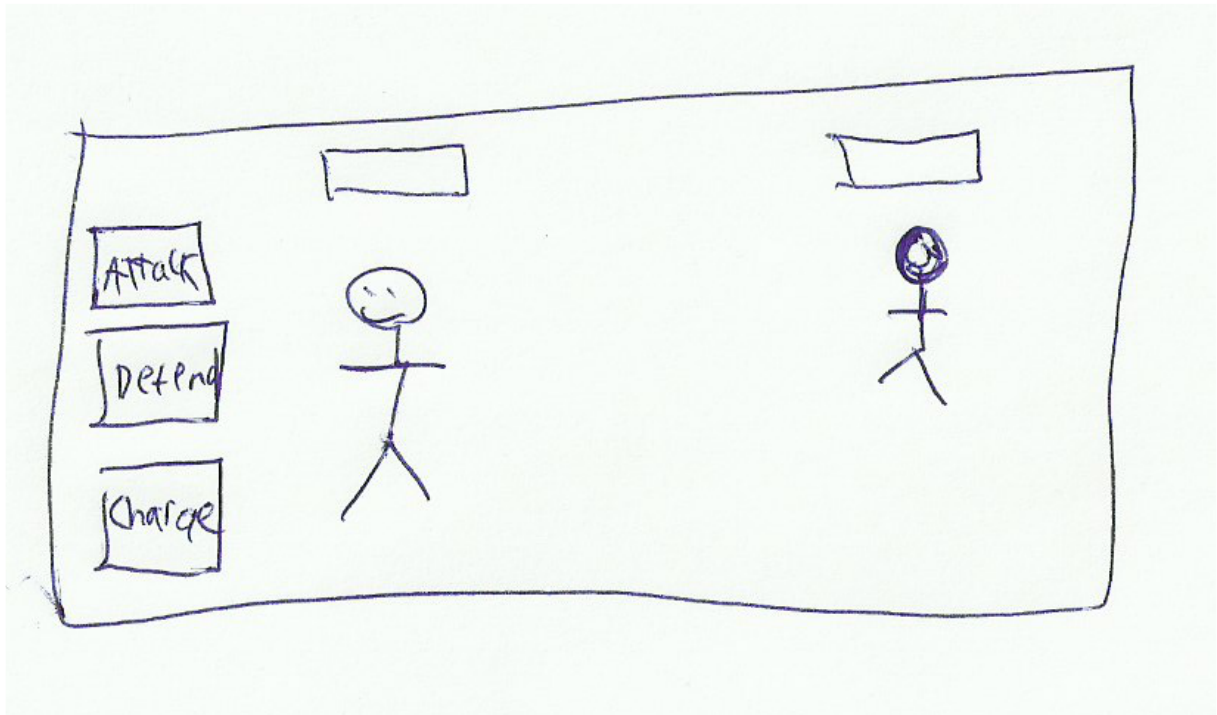
Email

PassWord

Team

Avatar Name

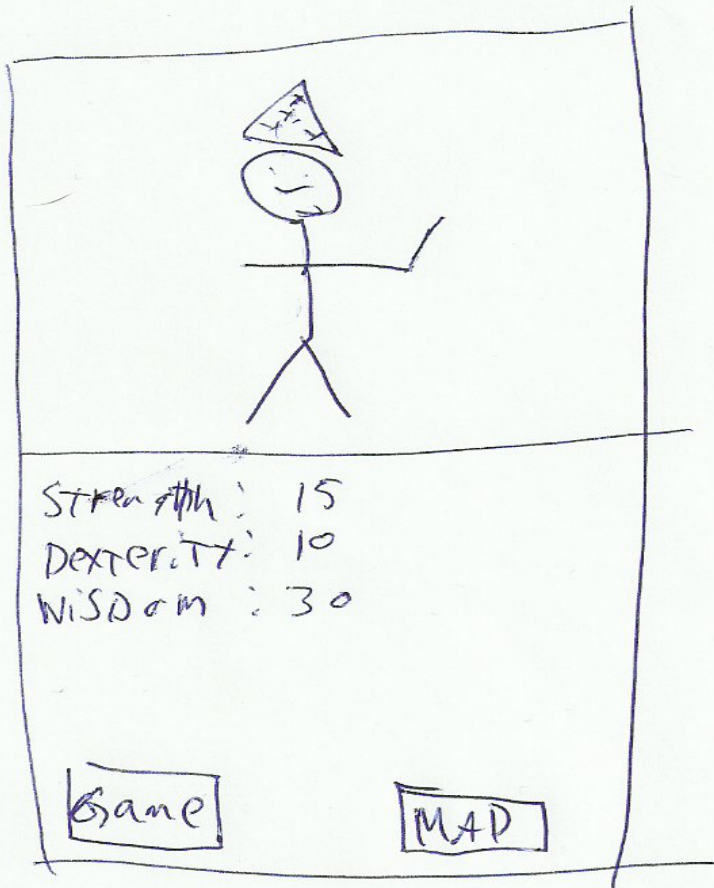
Battle scene:



Character screen:



# Character Screen



*Development*

---

## **Base classes**

1. Game controller (Singleton class)
  - a. Save/Load game state
  - b. Keep state between scenes
2. Guardian
  - a. All the characters statistics
  - b. Fight methods (lose HP, gain HP, Attack, block, die)
3. Map
  - a. Show the player a map of the local area and any relevant beacons
4. Server client class
  - a. Handles all communication with server
  - b. Has to support json encoding/decoding

## **Priority Schedule**

1. Implement server
2. Basic client that allows registration
3. Client that can show beacons placed on a map
4. Players can fight for and/or take ownership of beacons
5. Administration interface to place beacons
6. Indoor beacons
7. Guardian classes
8. Create new priority list for remaining tasks

## **Must haves**

- Players can choose a character
- Players can own beacons
- Players can fight other players for control of beacons
- Players able to register their character with a name
- Beacons can be placed outdoors

## **Should haves**

- Character classes
- Administration interface to place beacons
- players able to choose between different characters
- Beacons can be placed indoors'
- Possibility to create multiple instances of the game

## Would be nice to have

- Multiplayer chat
- Automatic package drop as item beacons
- players able to customize classes
- players able to customize their avatar
- Location aware graphics in battle screen

# Graphics

---

## Style Attributes

Cartoon style characters. Simple graphics.

Colour scheme:

Earthy colours:

Primary: Orange

Secondary: Blueish gray/Greyish brown

Tertiary: Muted yellow

Team colours: Red, blue

## Graphics Needed

1. Guardians  
Three classes, Scout/Rouge, warrior and wizard
2. Map  
Supplied by google maps. Need icons for beacons and player location
3. UI elements:
  - Buttons
  - Menu backgrounds
  - Fight scene backgrounds

# *Sounds/Music*

---

## **Sounds**

1. Effects
  - a. Guardian attack sound
  - b. Guardian hit sound
  - c. Guardian defend sound
2. Feedback
  - a. Button press
  - b. Scene transitions

## **Music**

**No music required**

# *Business plan*

The game is created to enable running several instances of the game. The long goal is for the game to be easily customizable. This allows the game to have a custom instance with a supplied set of variables that can be set by the person creating the instance. For this reason the game is meant to be short lived and is targeting events and any organization that can use it for team building or for helping players getting familiar with a certain area.

Examples include: Town councils; making people more active and giving a cheap way of instigating people to spend more times outdoor.

Company events: A company can create an instance of the game and run a competition between its employees as a team building exercise

A university to help new student get familiar with campus and making it easier for student to find their way around campus

These are just examples, and there can be other uses too. Since the GPS beacons are virtual, and don't have to be placed physically, it would be possible to extend the game to work for boats with beacons at sea or for planes, with beacons in the air. Since the game also supports BLE, it is possible to place beacons inside buildings, making it possible to hide them in different rooms.



## Appendix B

### High Level concept

Earth guardians



# GUARDIANS OF EARTH



# PREMISE

- Two teams fight for control
- all players have one guardian
- Guardians can guard posts
- Points awarded for guarding a post

# POSTS

- The posts are placed in difficult to reach places
- On top of mountains
- In the forrest
- Hidden inside buildings

# PLAYERS

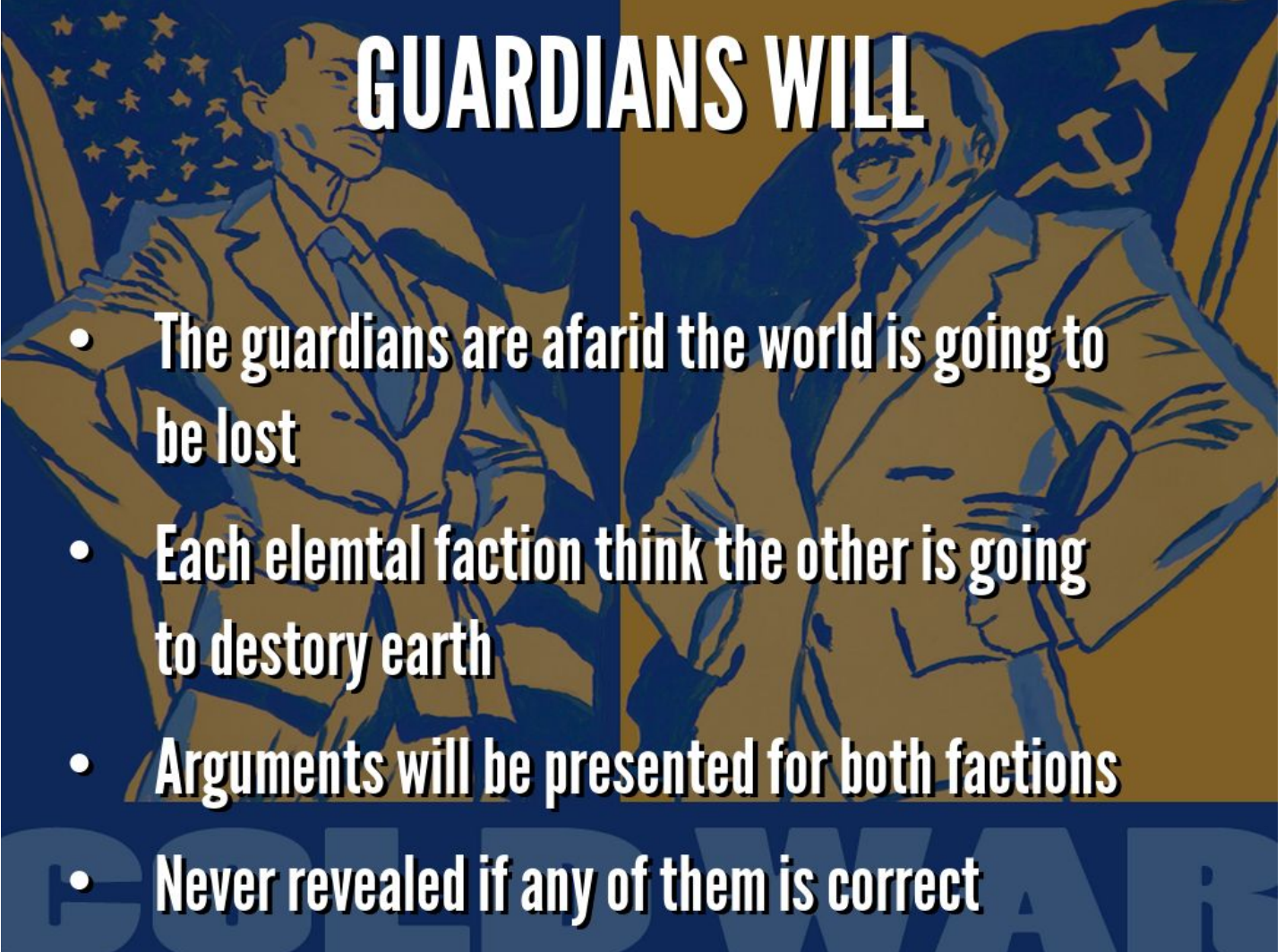
- Have a guardian container on their phone
- Guardians can only recover while in their container
- Containers are the only way to transport guardians

# GUARDIANS

- Guardians can't move around the earth by themselves (they need help)
- They have levels and gain experience for winning fights and finding hidden posts
- Can only heal when inside their capsule



Why fight?



# GUARDIANS WILL

- The guardians are afraid the world is going to be lost
- Each elemental faction think the other is going to destroy earth
- Arguments will be presented for both factions
- Never revealed if any of them is correct

# COLD WAR

# CONTROL

- A team controls a post when they have a guardian there
- Players from other teams can fight monsters left at a post. If they win they take control

# DOMINATION

- The game is played until one team owns all the posts
- Points are calculated every midnight
- If no one dominates all posts by XX/YY the team with the most points win
- The game will reset after one team has won





# MARKET POSITIONING

- Not made for profit
- Made for an event
- Event will decide length of game

# REVENUE

- Revenue will be based on selling to events or interested parties (councils, team building business)
- Only expenditure when game is created is server upkeep

# TARGET PLATFORMS

- **Strictly mobile**
- **Requires GPS and Bluetooth Low Energy**
- **Distribution will be through mobile markets**

# Bibliography

- [1] E. Adams and A. Rollings. *Fundamentals of Game Design*. Game design and development. Pearson Prentice Hall, 2007. ISBN 9780131687479. URL <http://books.google.no/books?id=jHghAQAATAAJ>.
- [2] S Balraj. Lte direct overview. *Qualcomm Research, San Diego*, 2012.
- [3] Richard Bartle. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research*, 1(1):19, 1996.
- [4] Entertainmen Software Association (ESA). Essential facts about the computer and video game industry. *None*, 20014.
- [5] V Fung, K So, E Park, A Ho, J Shaffer, E Chan, and M Gomez. The utility of a video game system in rehabilitation of burn and nonburn patients: a survey among occupational therapy and physiotherapy practitioners. *Journal of burn care & research: official publication of the American Burn Association*, 31(5):768–775, 2009.
- [6] Diana L Graf, Lauren V Pratt, Casey N Hester, and Kevin R Short. Playing active video games increases energy expenditure in children. *Pediatrics*, 124(2):534–540, 2009.
- [7] Viviane HASSELMANN and Peter OESCH. Are serious games promoting mobility an attractive alternative to converntional self-training for elderly people?
- [8] J.S. Holloway, D.J. Castor, O. Tsiupka, D. Kobylorkha, T.A. Mayes, A.M. Hunter, C.T. Spotts, and B. Hensel. Role based game play on a social network, June 7 2011. URL <https://www.google.com/patents/US7955175>. US Patent 7,955,175.
- [9] Emily A Holmes, Ella L James, Thomas Coode-Bate, and Catherine Deepro. Can playing the computer game "tetris" reduce the build-up of flashbacks for trauma? a proposal from cognitive science. *PloS one*, 4(1): e4153, 2009.
- [10] Carl Gustav Jung. *The archetypes and the collective unconscious*. Routledge, 2014.
- [11] Steven Kent. *Could You Repeat That Two More Times?* Three Rivers Press, 2001.

- 
- [12] Auguste Kerckhoffs. *La cryptographie militaire*. University Microfilms, 1978.
- [13] Hannu Korhonen, Markus Montola, and Juha Arrasvuori. Understanding playful user experience through digital games. In *International Conference on Designing Pleasurable Products and Interfaces*, pages 274–285, 2009.
- [14] Jonas Lauritzen, Adolfo Muñoz, Jose Luis Sevillano, and Anton Civit. The usefulness of activity trackers in elderly with reduced mobility: a case study. In *MedInfo*, pages 759–762, 2013.
- [15] Jørn Lomax. Optimizing games for mobile platforms. Capstone project Fall 2014.
- [16] Mark Masse. *REST API design rulebook*. ” O’Reilly Media, Inc.”, 2011.
- [17] Clifford Mayes. Cultivating spiritual reflectivity in teachers. *Teacher Education Quarterly*, pages 5–22, 2001.
- [18] Clifford Mayes. A transpersonal model for teacher reflectivity. *Journal of Curriculum Studies*, 33(4):477–493, 2001.
- [19] Cynthia L Ogden, Margaret D Carroll, Brian K Kit, and Katherine M Flegal. Prevalence of obesity and trends in body mass index among us children and adolescents, 1999-2010. *Jama*, 307(5):483–490, 2012.
- [20] L. Rosenberg. Multiplayer gaming using gps-enabled portable gaming devices, August 16 2007. URL <https://www.google.com/patents/US20070190494>. US Patent App. 11/697,704.
- [21] Adrienne Shaw. What is video game culture? cultural studies and game studies. *Games and culture*, 5(4):403–424, 2010.
- [22] Ryan West. The psychology of security. *Commun. ACM*, 51(4):34–40, April 2008. ISSN 0001-0782. doi: 10.1145/1330311.1330320. URL <http://doi.acm.org/10.1145/1330311.1330320>.
- [23] Talmadge Wright, Eric Boria, and Paul Breidenbach. Creative player actions in fps online video games. *Game studies*, 2(2), 2002.