

Keepers of Intheris: Mechanics and Technology

MASTER PROJECT

Yuri Aristides da Silva Godinho de Almeida

MASTER IN COMPUTER ENGINEERING



UNIVERSIDADE da MADEIRA

A Nossa Universidade

www.uma.pt

February | 2017

Keepers of Intheris:
Mechanics and Technology
MASTER PROJECT

Yuri Aristides da Silva Godinho de Almeida
MASTER IN COMPUTER ENGINEERING

SUPERVISOR
Eduardo Leopoldo Fermé

CO-SUPERVISOR
Sergi Bermúdez i Badia



Keepers of Intheris: Mechanics and Technology **Yuri Aristides da Silva Godinho de Almeida**

Constituição do júri de provas públicas:

Karolina Baras, Prof.^a Auxiliar da Universidade da Madeira, Presidente

Pedro Campos, Prof. Auxiliar da Universidade da Madeira, Vogal

Sergi Bermúdez i Badia, Prof. Auxiliar da Universidade da Madeira, Vogal

Março 2017

Funchal – Portugal

Abstract

In recent years, the development of video games has become more and more affordable, with several game engines being released for free, which allowed small independent teams to produce quality games. In this project, we detail the development process of a digital game which is an adaptation of a previously existing board game. This project focuses on the development and technological implementation as well as the game mechanics, two of the four elements of the game design tetrad, which encapsulates every aspect existing in a video game. The existing board game is a turn based strategy game, with 6 heroes, 48 unique abilities and 55 element powers. Here we will explain how we designed the architecture of this project, which technologies we used and developed to allow us to achieve this adaptation, and how we structured the multiplayer architecture to allow networked play. This work provides a deep insight for anyone that wishes to understand the game industry and learn the process of creating a video game.

Resumo

Nos últimos anos o desenvolvimento de jogos de vídeo tornou-se cada vez mais acessível, com vários motores de jogo a serem lançados gratuitamente, o que tem permitido que equipas pequenas e independentes produzam jogos de qualidade. Neste projeto detalhamos o processo de construção de um jogo digital, a partir da adaptação de um jogo de tabuleiro previamente existente. O projeto foca-se no desenvolvimento e implementação tecnológica assim como das mecânicas do jogo, dois dos elementos da téttrade do desenho de jogos, que encapsula todos os aspetos existentes num jogo de vídeo. O jogo de tabuleiro existente é um jogo de estratégia por turnos, com 6 heróis, 48 habilidades únicas e 55 poderes de elementos. Iremos explicar como projetámos a arquitetura deste projeto, quais as tecnologias usadas e desenvolvidas para nos permitir alcançar essa adaptação, e como estruturámos a arquitetura multijogador para permitir jogar em rede. Este trabalho fornece assim uma visão sobre como este processo é criado, para que possa servir de orientação a quem desejar desenvolver o seu próprio jogo.

Keywords

Game Development; Board game; Digital game production pipeline; Unity3D; Database for digital games

Palavras-Chave

Desenvolvimento de jogos; Jogo de tabuleiro; Processo de produção de jogos digitais; Unity3D; Base de dados para jogos digitais

Acknowledgments

Em primeiro lugar começo em português, pois nem todas as pessoas da minha família sabem ler inglês e quero começar por agradecer a eles. Muito obrigado por todo o apoio e por nunca terem deixado de acreditar em mim, nem mesmo quando eu cheguei a duvidar de mim!

I want to thank everyone who believed and supported me during this project. It was an extremely hard and long process and I would not be able to have done this without the help and support of everyone involved.

A special thank you to my supervisors, Professor Eduardo Leopoldo Fermé and Professor Sergi Bermúdez i Badia, from Universidade da Madeira, for all the time they dedicated to this project. Without them, the final product would not have the same level of quality.

To my friend and co-worker, Tatiana Vieira, a very warm thank you. Your dedication and hard work were always a great source of inspiration for me and made me gave 110% all the time.

I would also like to thank João Serina. Your insight in the game design process and the help you gave us balancing the game was priceless!

To Juan Ponte, thank you for brightening our days with your peculiar sense of humour and for making the final version of the GUI look much better.

Finally, to all professors who taught me everything I know and gave me *carte blanche* to follow my dreams and always allowed me to ‘use’ their courses to learn more about video games.

To you my love, thank you for being my muse, my source of inspiration.
Thank you for making me a better man.

Contents

1	Motivation.....	3
2	State of the art	5
2.1	Gaming Industry	5
2.1.1	Game history	5
2.1.2	Strategy Video Games	6
2.1.3	Rise of strategy games	7
2.2	Video games development.....	8
2.2.1	Game Development pipeline	9
2.2.2	Game development vs software development.....	9
3	Prior work: Game of Pawns.....	11
3.1	Paper prototyping	11
3.2	First concept.....	11
3.3	Deviation of initial concept	12
3.3.1	Mechanics cut and added	12
3.4	Final concept	13
3.5	Final version gameplay.....	13
3.6	Game Requirements	15
4	Development process.....	21
4.1	Game Technologies	23
4.1.1	Game Engines.....	23
4.1.2	Database analysis	24
4.1.3	Networking.....	25
4.2	Implementation.....	28
4.2.1	Architecture.....	28
4.2.2	Networking.....	34
4.2.3	Game mechanics	37
5	Prototypes	47
5.1	Prototype 1.....	47
5.2	Prototype 2.....	47
5.3	Prototype 3.....	48
5.4	Final prototype	49
6	Playtests	53
7	Conclusion	55
8	References.....	57

9	Appendices	63
9.1	GANTT	63
9.2	Requirements	65
9.2.1	Game Abilities	65
9.2.2	Element Powers	69
9.3	Game Systems	73
9.4	Database.....	79
9.4.1	Account tables.....	79
9.4.2	Elements balancing table	80
9.4.3	Heroes and Abilities balancing tables	81
9.5	Bug reports.....	81
9.5.1	Playtest 23-11-2016	81
9.5.2	Playtest 28-11-2016	85
9.5.3	Playtest 02-12-2016	89
9.5.4	Playtest 06-12-2016	90
9.5.5	Playtest 24/25-12-2016.....	94

Figure Index

Figure 1 – Game division. Orange is the work done in GGP. Green is the work done by the DAP team. In blue is the work done by the MTP team and in white is what was not implemented. ...	3
Figure 2 – Nimatron. A: the pattern submitted for the machine. B: a woman demonstrates how to play the game. [10]	5
Figure 3 – A: an arcade machine for pong. B: an Atari 2600 Console. [12]	5
Figure 4 – Nintendo Entertainment System, the first console to require a licensing agreement to permit development of software on it. [16]	6
Figure 5 – League of Legends, the top grossing PC game of 2015. It generated 1.628 billion USD, 500 million more than the game in second place. [85]	6
Figure 6 – Red Alert 2, Fog of War feature being demonstrated with zones blacked out, zones greyed out and zones perfectly visible.....	7
Figure 7 – Standard layout of a MOBA game map.....	8
Figure 8 – Traditional job hierarchy in a game development studio [30].....	9
Figure 9 – First prototype of the board game. Notice the amount of space required to play this version.....	11
Figure 10 – Original element power concepts. Each element had two different behaviours, varying if it was the first or the second to be applied to the target.	12
Figure 11 – Final version of the board game prototype. Compare the amount of space required to play the game, compared with the first version, in Figure 9.....	13
Figure 12 – Demonstration of how elemental combinations work on our game. In this example, if the attacker (Fire) hits fire, it will generate a neutral combination. If it hits the adjacent elements to fire (earth or wind) will generate a positive combination. If it hits any non-adjacent element (lightning or water) then it will produce a negative combination.....	14
Figure 13 – Map Editor developed for us to create our map. The panel on the top right of the screen displays the properties of the selected tiles (blue tiles). It is possible to change these properties by clicking on the highlighted buttons.	22
Figure 14 – A: UE4 displaying all the properties of a cube at creation time. B: U3D showing all existing properties of a cube at creation time.....	24
Figure 15 – U3D's High-level API for network games. Uses their own low-level API to handle the most common requirements for multiplayer games [63].....	26
Figure 16 – Client-Server communication pipeline in u3D. Every action the client want to perform is sent to the server via a command. The server process the data and send information back to the client through a RPC [50].	27
Figure 17 – Relationships between the user account and their possessions and achievements in the database. Full version can be found in appendix 9.4.1.	29
Figure 18 - EP, Hero and Abilities entities in the database and their relationship. Full version of this section of the database can be found in appendix 9.4.2 and 9.4.3.	29
Figure 19 – HeroCore class and all its properties.....	30
Figure 20 – Ingame hero with all its properties and a reference to the HeroCore.....	31
Figure 21 – Complete hero hierarchy and the relations each type of hero has with other heroes.	31
Figure 22 – Division for MVC implementation. On the left, there are the Models. On the right, there are the Views and in the middle the Controllers.....	32
Figure 23 – Implementations of the database abstraction. Blue represents what was developed in this project, white was implemented in the other project	33

Figure 24 – Enum set in u3D Editor to choose which database handler to use by the factory. .	33
Figure 25 – Code snippet for the singleton implementation in U3D	34
Figure 26 – Code snippet for a Command. This code is executed only on the server side of UNet code and can only receive primitive types or arrays of primitive types as parameters.	34
Figure 27 – Flowchart of server side validation. Client send the Command with relevant parameters to the server. Server verifies if those parameters are valid for the action the client wants to perform and answer accordingly by sending an RPC.....	35
Figure 28 – Flowchart of the network pipeline using Node.js. It is very similar to UNet pipeline.	36
Figure 29 – Socket.IO component with the URL pointing to the server location. We could use localhost for testing and then deploy remotely to our Amazon EC2 machine.	36
Figure 30 – A: offset coordinates. B: axial coordinates. C: Conversion from offset coordinate system (represented by X and Y) to axial coordinate system (represented by Q and R).	37
Figure 31 – Map editor tools developed to allow us to quickly change information about the map and the tiles. It can be used to add rows and columns to the extremities of the map and to change tile properties. Data is then stored in a Scriptable Object and is read and rendered in the game.....	38
Figure 32 – Lipp executing Lipp2. A: Lipp swaps position with its target. B: Lipp starts animation to push secondary targets away from her. C: Targets land on the first empty available tile for them and return to their idle animation.....	41
Figure 33 – Execute Ability pipeline. After validating all conditions for the usage of an ability, the game check if the ability will be executed normally or with exceptions.	42
Figure 34 – Demonstration of using an ability (Lipp2). Selected hero is Lipp, from team Chorinis. It displays all information relevant to the currently selected ability and informs the player that he can use it.	45
Figure 35 – Extrapolation lines generated for each secondary target of Lipp 2 abilities. these lines pass through as many tiles as the 'distance' parameter received in function ExtrapolateHex. .	45
Figure 36 – Code for extrapolating the first available tile in a given distance line.	46
Figure 37 – Execution of the EP gained after the execution of the ability Lipp2.....	46
Figure 38 – Low fidelity prototype of the ability system. This was the second prototype of the game and it could be used to test abilities without the existence of a map.	48
Figure 39 – Prototype that allowed us to implement and test all game mechanics in the network.	49
Figure 40 – Menus scenes with functionality implemented. A – Login screen. It is possible to login and create a new account from this screen. B – Shop screen. In the shop the player can buy heroes still locked in his account. Heroes are bought with in-game currency. C – Team Selection screen. From here, the player can choose all heroes in a team and change their element. D – Find a Game screen. In this prototype, only the Versus game mode was implemented.	50
Figure 41 - Final prototype with all mechanics from the GGP implemented.	50
Figure 42 – Actions that can be performed by the game state altering feature.	51
Figure 43 – Game Cycle flow chart.....	73
Figure 44 – New round and Choose Elemental Power Zone element flowcharts.	73
Figure 45 – Hero turn flowchart.....	74
Figure 46 – Hero attack Objective flowchart	75
Figure 47 – Gain Element Power flowchart	75
Figure 48 – Verify can execute ability flowchart.....	76
Figure 49 – Execute Ability flowchart.....	77
Figure 50 – Use ability flowchart.....	78

Figure 51 – Unlock Ability flowchart 78

Figure 52 – Account tables with all possessions and achievements..... 79

Figure 53 – Elements and Element Power balancing tables 80

Figure 54 – Heroes and Abilities balancing tables. 81

ACRONYMS

CE5 – Cry Engine 5
UE4 – Unreal Engine 4
DAP – Design and Aesthetics Project
DotA – Defence of the Ancients
EP – Element Power
FoW – Fog of War
GGP – General Game Project
GIP – General Implementation Project
GPU – Graphical Processing Unit
GUI – Graphical User Interface
LINQ – Language Integrated Query
LoL – League of Legends
MOBA – Massive Online Battle Arena
MTP – Mechanics and Technologies project
RDBMS – Relational Database Management System
RTS – Real Time Strategy
U3D – Unity3D Engine
UNet – Unity Networking

PART I: INTRODUCTION

1 Motivation

This project follows the work we developed for our Bachelor Degree's final project in Computer Engineering at Universidade da Madeira, which will be referred as the General Game Project (GGP). This work, developed by two students, Tatiana Vieira and Yuri Almeida, had the goal of create a board game to validate a game concept and to extract all the requirements needed for a digital implementation. The digital implementation, referred as the Game Implementation Project (GIP) was vast and complex and, as such, it was divided in two subprojects. The Design and Aesthetics Project (DAP [1]) focused on the Story and Aesthetics, while the Mechanics and Technology Project (MTP), focused on the mechanics and technologies involved in the game.

The board game developed on the GGP was a three vs three strategy board game, with great emphasis on teamplay, positioning and hero synergy. The teams faced each other in the battlefield with the objective of destroying each other's base. The gameplay was slow and methodical.

Some adaptations had to be made for the GIP, the main one being that the game is now one vs one, where each player controls three heroes. The game is multiplayer and is designed to accommodate a free to play business model. There is a shop where players can buy more content. This content can be bought with in-game currency, that the player receives as a reward for completing objectives in the game, or with real world currency.

Following the Elemental Tetrad of Games [2], the GIP was divided in two. It was our responsibility to design the architecture of the project, including the database, data structures and networking. We also had to decide if we were going to use an existing game engine, or if we would develop our own. Additionally, we had to choose a database engine and networking system to use. We also had to translate all the mechanics from the board game to the digital game (Figure 1).

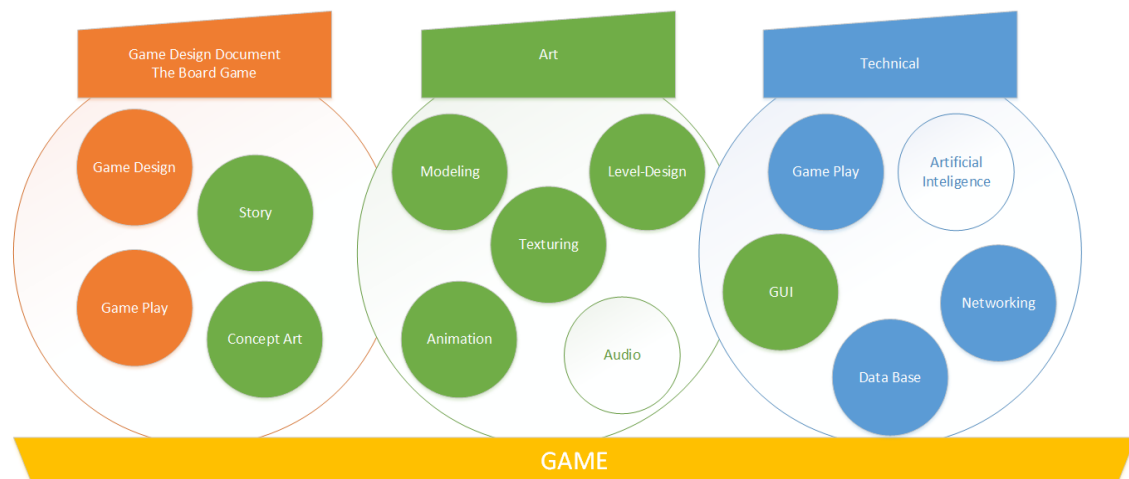


Figure 1 – Game division. Orange is the work done in GGP. Green is the work done by the DAP team. In blue is the work done by the MTP team and in white is what was not implemented.

In this report, we start to show a brief description of the Video Game Industry, its influence in our world and some of the most influential titles. We then explain in more detail how the GGP was developed and how to play that version of the game. After that, we proceed to describe the development process, including the technologies used and how it was implemented. To conclude, we display prototypes created and the playtests performed during the development.

2 State of the art

2.1 Gaming Industry

The gaming industry is one of the fastest growing industries [3]. With a prediction of 91.8 billion US Dollars of revenue in 2016 [4], the gaming market is very attractive. In Europe, there are five major market pockets: Germany, UK, France, Spain and Italy. The European market has around 337 million gamers, which corresponds to approximately 45% of the European population. The North American market is very similar, with 55% of the population playing video games [4]. With digital distribution being the main method of reaching gamers in the past few years, both of these markets are accessible from anywhere in the world [5].

2.1.1 Game history

The gaming industry is a driving factor for the evolution of personal computers: sound cards, graphic cards and 3D accelerators were all greatly improved due to the requirements of games. The first dedicated Graphical Processing Units (GPU) was developed to allow the rendering of more colours. Later, the support to render an overlay layer was added, allowing developers to implement Graphical User Interfaces (GUI) [6], [7].

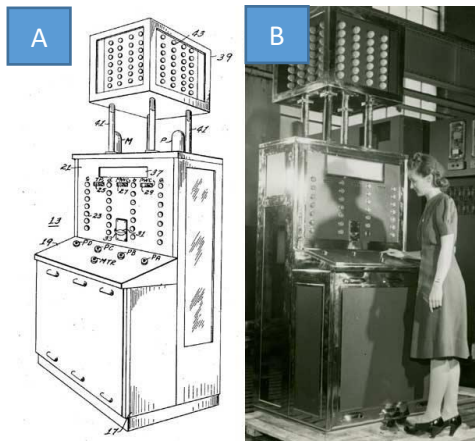


Figure 2 – Nimatron. A: the pattern submitted for the machine. B: a woman demonstrates how to play the game. [10]

The gaming industry had its birth in the early 40's, showing up at science fairs as an oddity. The first gaming machine that could be recognized as one was unveiled at the New York World Fair in 1940, by Dr. Edward Condon (Figure 2). The machine played an ancient mathematical game, called Nim. During the six months it was available at the fair, around 100,000 games were played with the machine winning 90,000 of them [8]–[10].

Other machines like these were popping up in fairs here and there, but it was not until 1967 that the first home system was designed by Ralph Bear [9]. The system was licenced for manufacturing to Magnavox, under the name of Magnavox Odyssey which sold

350.000 units during its life time [11].

In 1972, Atari released the arcade game Pong, which was a huge success worldwide. Five years later, Atari condensed the hardware into a small box, called it Atari 2600, and sold the Pong game to their first home console (Figure 3). Although the Magnavox Odyssey sold very well, it was not until the Atari 2600 that the gaming industry exploded, with Atari selling over thirty million units of this machine during its lifespan, 24 years [12].

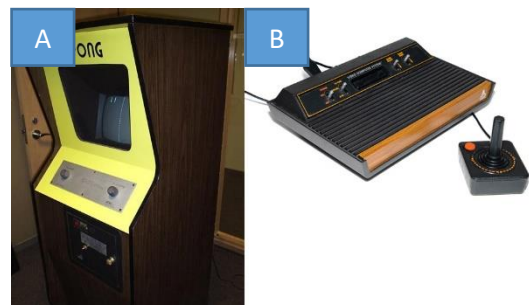


Figure 3 – A: an arcade machine for pong. B: an Atari 2600 Console. [12]

In 1982, a law was passed in the United States, making it legal to develop games for Atari 2600 as a third-party company. This meant that everyone could develop and sell games on any of the

existing consoles. And everyone did. Due to this lack of licencing regulation, the market was flooded with low quality games, which dropped their prices. A new game, which used to cost 60 US dollars, was now being sold for 2 or 3 dollars. Since most low-cost games were also of low quality, the public lost confidence on the market. This led to a market crash that almost destroyed the gaming industry [13].

The greatest consequences of this crash were the development of a licensing agreement to develop to consoles, which allowed the manufacturers to guarantee a certain level of quality on any product sold to their device, and the development of anti-piracy hardware to be installed in the consoles. It is important to note that this crisis had little to no impact in the computer gaming market [13], [14].

The first manufacturer to put this new agreement in practice was Nintendo, with the NES console (Nintendo Entertainment System, called Famicom on Japan) [15] (Figure 4). Nintendo had oversight of what was developed for their console and they could deny any low-quality game from being published. This gave the public more confidence on what they were buying. The market started recovering at this point and never recessed again [16].



Figure 4 – Nintendo Entertainment System, the first console to require a licensing agreement to permit development of software on it. [16]

The market has been in an ever-growing cycle since the '83 crisis and in 2015 reached 91.8 billion dollars worldwide [17], more than half being on digital sales [18]. The Pc market was always very attractive, and it has finally surpassed the consoles and mobile markets combined on 2015 [19].



Figure 5 – League of Legends, the top grossing PC game of 2015. It generated 1.628 billion USD, 500 million more than the game in second place. [85]

In 2008, for the first time in history and every year after this, the gaming industry generated more revenue than the movie industry [20], and in 2015, the difference was almost three times in favour of the gaming industry [17], [21]. The gaming industry is a very attractive market, especially the free to play market on PC with seven of the ten most grossing games of 2015 being free to play [22] (Figure 5).

2.1.2 Strategy Video Games

Strategy games has always been one of the most popular genres on the PC market. These games focus more on the intellectual capability of the player than on his physical skills. It emphasizes strategical, tactical and logistical challenges. Economic challenges and exploration are also usually featured on this genre. There are four main sub-types of strategy video games: Real-time or Turn-base games and whether the game focus on strategy or tactics. The best players are those who can plan a series of actions against one or more opponents and execute them to perfection to achieve their victory. The destruction of enemy forces is usually a main goal on this genre and victory is achieved through superior planning, delegating luck and chance to a secondary role [23], [24].

Strategy games can be organized in two big groups: real time and turn based. Both have many subgenres and it is usual that a game be part of more than one sub-genre. Even though on this work we will focus more on the turn based facet of this games, it is also important to talk about the rise of real time games that led to the birth of the Multiplayer Online Battle Arena (MOBA) and their influences on our game.

2.1.2.1 *Gameplay*

These games usually oppose two teams in a map with the objective of destroying each other's base. Each team can be made up by one or more players, but both teams generally have the same number of players. The gameplay revolves around gathering resources to build an army to destroy the opponent, while defending against their attack.

The map usually features Fog of War (FoW), which hides parts of the map that do not have the player's unities and/or structures (Figure 6). This FoW is usually fully opaque until the player explores it for the first time, then it will be greyed out if no units belonging to that player are nearby. In the greyed-out area, only static elements are visible.



Figure 6 – Red Alert 2, Fog of War feature being demonstrated with zones blacked out, zones greyed out and zones perfectly visible.

The game ends when the opponent is defeated, i.e., if he has no more structures, or when one of the players leaves the game.

2.1.3 Rise of strategy games

These games were heavily inspired by traditional board games, like Chess and miniature wargames, such as Risk [25]. In fact, the first strategy game was a Risk-like game called Invasion, released in 1972 for the Magnavox Odyssey. Several other titles were released in the following decade and in 1983 the first 4X (explore, expand, exploit and exterminate) game was published by SSG [26], which expanded the concepts of economic growth, technological progress and conquest. By the early 90's, strategy games were a popular genre and the first real time strategy games started to appear. In 1992, Westwood Studios [27] published Dune II, an RTS based on Frank Herbert's science fiction novel [28]. Dune, alongside Warcraft: Orcs & Humans and Command & Conquer, started to dominate the video game market. By the early 2000's, strategy (either real-time or turn-based) was the most popular genre, with titles like Warcraft III: Reign of Chaos, StarCraft and Civilization III leading the sales in the gaming market.

2.1.3.1 *Birth of MOBAs*

One of the main differences between Warcraft or StarCraft and the other RTS's of the time, was the powerful "World Editor" developed by Blizzard Entertainment. This editor allowed players to completely remake the game, instead of simply creating new maps with the existing concepts and assets. The modifications (MODs) done by the players could then be bundled in new game modes and shared with the community of that game. This led to several extremely creative MODs, which, in time, evolved into their own game genre. Two of the most known were Tower Defence, and the ancestor of MOBAs, Defence of the Ancients (DotA). This MOD defined the pillars of the design for this genre, including the map layout and the several game concepts present in all games of this genre today (Figure 7).

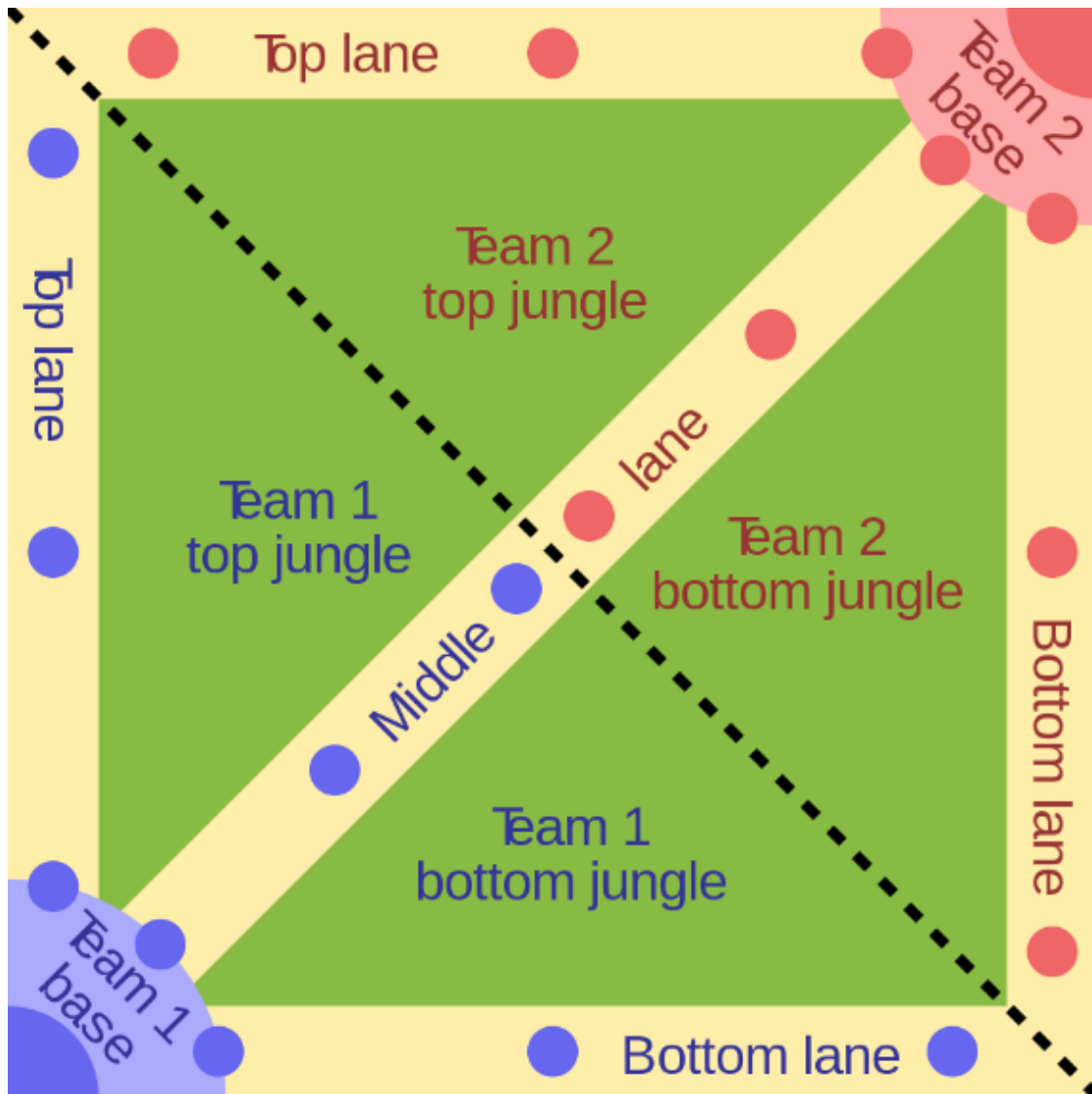


Figure 7 – Standard layout of a MOBA game map.

2.1.3.2 Rise of MOBAs as dominant computer game genre

By 2008 this genre started to attract commercial attention and in 2009 Riot Games released League of Legends (LoL), a game very similar to DotA, but unlike its muse, LoL was not a MOD and it was free to play. The gaming community started to flock to this title and soon it became the most played video game in history. In 2013 Valve released DotA 2, a free to play standalone updated version of DotA. In 2015, Blizzard Entertainment, the company responsible for releasing Warcraft III, released their own MOBA called Heroes of the Storm, based on the heroes of their three major universes (Diablo, Warcraft and StarCraft). The reception for all these games was extremely positive and this game genre became the most played, by far, with League of Legends hitting the 100 million unique monthly users mark in September of 2016 [29].

2.2 Video games development

As previously mentioned, video games have a huge market share and it is very likely that this market will keep growing in the foreseeable future. Considering this, it is important to know how video games are made and how do this development differs from traditional software development. We will focus on the development of 3D games.

2.2.1 Game Development pipeline

One thing that it is important to know about game development, is that the same role might have very different skills, requirements or responsibilities in different game studios. However, there are some standardizations around developing a video game. There are usually four major areas of expertise in a game studio: design, art, animation and programming (Figure 8).

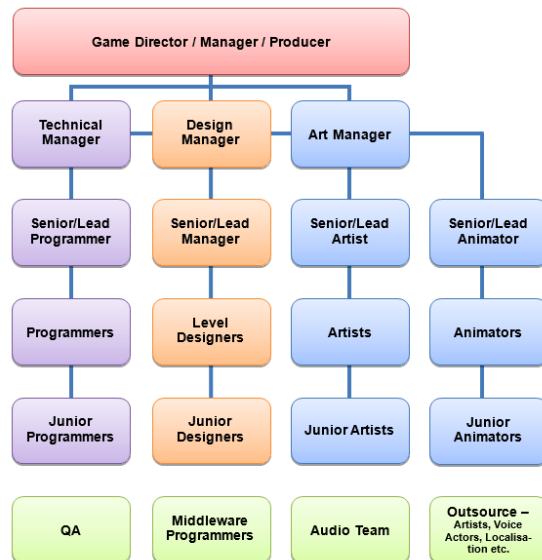


Figure 8 – Traditional job hierarchy in a game development studio [30].

At an early stage, designers are responsible for writing and creating the bases for the game: what does the player must accomplish, which are the game's goals, what are the obstacles, will the player have help from other players or characters, will the game be an FPS, an RTS and from which existing games can we draw inspiration. All designers, from game to graphical designers, will take part in this process.

After this process, it is time to build some prototypes: gameplay mock-ups, technology prototypes, paper prototypes of the user interface or sketches of characters and locations. This will help define a visual language and gather requirements for the game development later.

Once the concept art is developed, then the 3D artists can start to model the characters and the world. This process is extensive and explained in detail in the DAP's manuscript [1]. In short, a 3D model must be built, then a process to reduce the polygon count is applied so it can be textured and animated and finally integrated into the game engine. At this time, visual effects artists and the audio designers can also start to produce content that will be used in the game later.

In parallel to this work, the programmers will develop all the gameplay related aspects of the game. All the systems existing in the game, such as an achievement system or a progression system, must be programmed. Content for these systems is not important for now, just that the system exists and that they can be used later in the game.

Once most of the content for the game is developed, it is then time for the level designers to assemble all the developed systems to create the proper game: when will the player need to jump over an obstacle, which abilities must the player use to defeat this boss and which skillsets can be used to open this door.

Finally, the game is complete but, now it is time to fix all the bugs and improve performance of the game. This is where the Quality Assurance (QA) department comes to action. Their job is to identify every problem with the game and forward a report to the respective department [30], [31].

2.2.2 Game development vs software development

One thing that makes itself evident in the previous section is that a pipeline for game development does not differ much from regular software development: you start with a concept

and low cost mock-ups of what you want to develop and you build on top of that until you have a product that fulfils all the requirements.

In software development, the testing is done to show that the software does what it is intended to do and discover all bugs before releasing the program. Usually these tests are organized in three stages: Development testing, Release testing and User testing. Development tests are performed by the development team to find defects and bugs. Release tests are performed in the complete version of the software and it is tested by a separate team before being released to the final users. The main goal of this test phase is to verify that the software meets all the system requirements specified by the stakeholders. The final phase, the User testing, is performed by potential users of the system in their own environment. Even if the system has been thoroughly tested, the user testing is always essential. Changes made at this time, based on the user feedback, can have a major effect on reliability, usability, robustness and performance of the software [32].

In game development, besides doing all the tests mentioned for software testing, there is some additional testing that needs to be performed. We can divide this extra testing phase in three stages: *first playable* version, *alpha* version and *beta* version. The first playable version, as the name indicates, is the first version of the game where all the major gameplay elements are implemented and playable. At this stage, the game must be essentially fun, it should be easy for the team to play and start to quickly brainstorming off new features to the game. The main goal when starting to develop a game is to reach the first playable version as soon as possible. Most of the risks and doubts about the game should have been dealt with when reaching this stage. Once the development team finishes adding features to the game, then it has reached the *alpha* phase. At this point, the team should not be breaking the code to add new features, but rather cleaning up the code base. The game will still have bugs and balancing issues, but when reaching *alpha*, the game must be fun to play. If it is not fun, it might need more (or less) features. Between *alpha* and *beta*, the development team should be making the game more stable, fixing game crash bugs, text bugs and balancing the game. Usually, the *beta* version is described as a feature-complete product with no known game breaking bugs. At this point, the development team's focus should be on polishing the game. Adding new features is extremely dangerous, as it could lead to feature creep (the never-ending addition of new features to a game [33]). Once the number of bugs discovered by the development and QA teams are practically non-existing on each play sessions, then the game has reached the *beta* and can now start to be shipped to the beta testers. Once the number of bugs found by the users is minimal, the game is ready to release.

In regular software development, the developer starts by hearing the client and gathering the software requirements directly from him. Then, they start developing and are done once the software satisfies the needs of the client. In game development, you usually do not have a client and as such, you do not have software requirements to follow. The game is not done once you complemented your checklist, but rather when it is fun to play and 'fun' can be a rather difficult concept to describe in a technical way [34]. There are other differences: most software does not require 3D rendering, advanced shaders, physics engine or artificial intelligence. But the fun aspect of games is what makes it the most different. In this project, we had the system requirements, since we had previously developed a board game prototype before starting the production of the computer game.

3 Prior work: Game of Pawns

As previously mentioned, before starting the production of the computer game, we made a board game to prototype our ideas for the game. This served two purposes: first, it was an effective low cost prototype for the project, where we could test and iterate the game, and second, once completed, we could use it to gather the requirements for the development of the computer game.

3.1 Paper prototyping

This prototype was used to develop and balance all the game mechanics. Several playtests sessions were made to help us determine the game's viability and to balance all the mechanics. During this period, we also gathered all the requirements for the game.

By using this approach, we could do many iterations on our product at a very low cost. Changing an ability's mechanic was as simple as changing the text on a piece of paper.

3.2 First concept

We started by attempting to create our own version of a MOBA game, with all its components, on a board game. There were minions to move, levels to gain and stats to distribute. Added to this, we had the Elements mechanic, a new addition to the genre and unique to our game. All of this meant that the player had to perform a huge number of maintenance actions each turn: they had to move their minions, they had to track experience, cooldowns, buffs, debuffs, use and regain resources.

We tested all these mechanics individually and they seemed to take an acceptable amount of time to perform. It was not until we did our first internal play session that we realised the mistakes we had made. The game was daunting. After nearly four hours of play time, we had barely started the game and not even one of the objectives was destroyed.

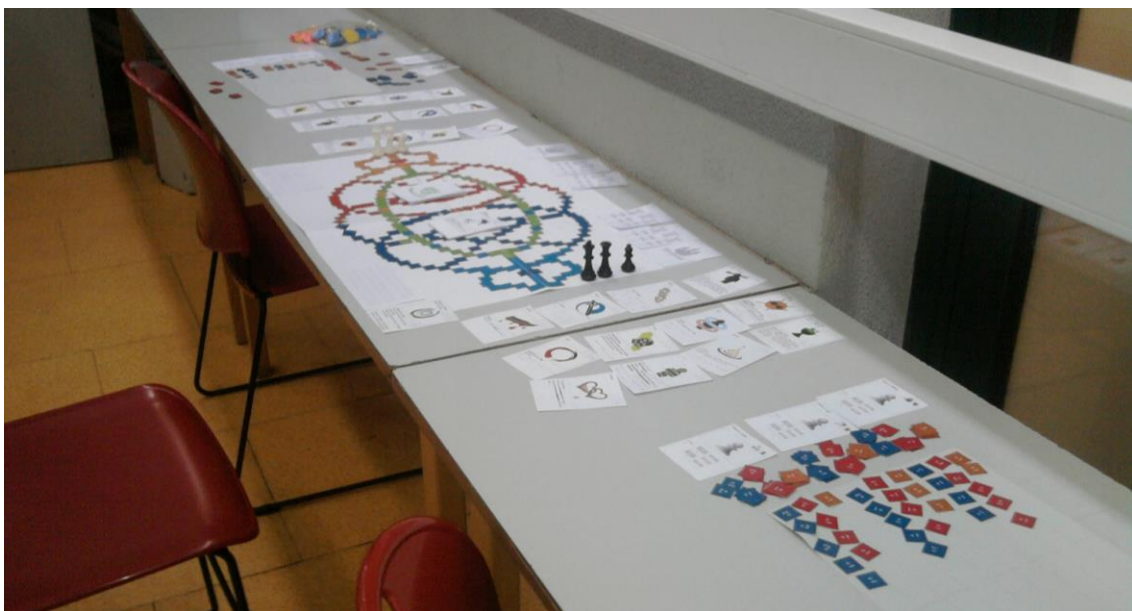


Figure 9 – First prototype of the board game. Notice the amount of space required to play this version.

3.3 Deviation of initial concept

3.3.1 Mechanics cut and added

Something had to be done about this, so we went back to the drawing board and started brainstorming again. There were too many systems that we felt added a lot of complexity to the game, without the benefit of adding any depth. In other words, many of the systems were a burden that did not add strategy, variability or any kind of replay value. We decided to remove these systems and focus only on what we considered was essential for the game to exist and then we would add new mechanics.

Experience points, which were gained by killing minions and heroes, were completely removed from the game. We decided that what was important was the hero (player) progression and not to count how many minions each player had killed. By removing the experience, the levels and minions were also cut, since they were part of the same system. This also led to the removal of individual stat management, where the player could increase the power of each aspect of their hero (health, damage or resources) as they levelled up.

Currency, which was gained by killing other heroes and/or objectives, was also removed, as well as all the items it bought. We felt that those items, especially the health potions, were only delaying the game, making it last longer.

Certain abilities and elements would apply effects on the target: stuns, snares, roots and many others. These effects were also removed from the game.

At that time, the element power mechanics were also very complex: the element the player controlled had different effects depending on when they were applied (Figure 10).

Water - 2nd Applied element					Wind - 2nd Applied element				
1st applied element	1st element:	Wind	Fire	Nature	1st applied element	1st element:	Water	Fire	Nature
gains 1 magic shield	Alies	1HP + 1HP for each target's ally at 2AOE range	HoT 1HP / 3 turns	1HP 1 AoE	+1 range in the next ability	Alies	2 RC	RC/4 current	1RC 1AoE
receives -1 HP/MS in the next Heal for 1 ronda	Enemies	-1HP e -1HP for each target's ally in 2AOE range	DoT 1 HP / 3 turns	-1HP 1AoE	-1 range in the next ability	Enemies	-2 RC	-RC/4 current	-1RC 1AoE
Fire - 2nd Applied element					Nature - 2nd Applied element				
1st applied element	1st element:	Water	Wind	Nature	1st applied element	1st element:	Water	Wind	Fire
1HP	Alies	1DMG for 4HP total	1DMG for 2RC total	1DMG for each cooldown timer on the CC	Caster gains immunity to CC during 1 round	Alies	Imune to stun	imune to root	imune to silence
1DMG	Enemies	-1DMG for 4HP total	-1DMG for 2RC total	-1DMG for each cooldown timer on the CC	-1 movimiento prox fase movimiento	Enemies	Stun	Root	Silence

Figure 10 – Original element power concepts. Each element had two different behaviours, varying if it was the first or the second to be applied to the target.

As illustrated above, this system was extremely complex with many different concepts for the player to memorize. This system was taken down and replaced by a less complex system where each element would have a theme.

3.4 Final concept

As we kept removing and changing more and more mechanics, we were getting further away from MOBAs. We ended up with a turn based strategy game, but kept our influence from the MOBAs. We focused more on the strategic and tactical aspects of the game rather than giving emphasis on the player's reflexes and agility. The effort to reduce complexity and improve depth paid off and we built a game that allows several winning strategies. Also, from our hundreds of hours of playtests, not even a single dominant strategy emerged, i.e., there is no single strategy that is better than all others in every scenario. Team work, hero and element synergy are rewarded and incentivised. The final version is displayed in Figure 11.



Figure 11 – Final version of the board game prototype. Compare the amount of space required to play the game, compared with the first version, in Figure 9.

3.5 Final version gameplay

Like the MOBAs, our game opposed two teams on a battlefield with the objective of destructing each other's base. Each team controls three heroes and each hero has eight abilities to choose from. Each hero must also choose one of the five Elements to control during the game.

We created six different heroes, fitting the common tropes of the fantasy archetypes: a brave Guardian who protects their allies, a cunning Pathfinder who keeps the distance and uses the terrain to her advantage, a bright Wizard who can be as devastating as controlling, an Assassin

who can kill before being seen and a Mender who can manipulate the life-force either to heal or kill.

As mentioned before, each hero has eight abilities. There are three categories of abilities: Utility abilities, which are used to defend or move the heroes, Damage abilities, which are used to damage their targets and Ultra abilities, which are the signature move of that hero. Each hero starts the game with one ability and each time their team kills an enemy or destroys an objective, they unlock a new ability, up to a maximum of four abilities per hero. The slots where the heroes can place their abilities have pre-determined Polarities. By giving the player more abilities than they can use, we force them to choose and to think of different strategies and ways to play with each hero. This also increased the amount of possible combinations and gives them almost unlimited variability.

The five Elements chosen for the game were: Water, which helps the player keep their team alive, Fire, which is used to damage the opposing team, Wind is used to move heroes around and to defend the Objectives while Earth is used to attack them. Lightning changes the game rules temporarily and is the most volatile element. Each element has eleven different powers with three levels of rarity on the deck: common, rare and epic. There are three of each common card, two rare and one epic. There is a total of 25 cards per deck.

These Elemental powers are the only luck element in our game, since the player does not know which card they will get before drawing. They can, however, be assured that a drawn power is a positive action and they can control who draws them. This keeps the luck to a minimum and lets the player decide what will happen, improving the strategy and tactic of the game and at the same time, keeps a small luck factor in the game that can help to turn the tables and give the losing player a fighting chance.

Each time a hero uses an ability, he applies his Element to the target. This element interacts with the previous applied element to produce one of three things: a positive, negative or neutral combination. Neutral combinations count as a positive and a negative at the same time. At the end of the turn, the player must count how many combinations were performed. If there were more positive combinations, his team draws one card of that hero's Element. If there were more negative, the opposing team draws the card instead. If the number of positives and negatives is the same, then both teams draw a card with the playing team drawing first. This is what we call the combination systems.

By combining adjacent elements, the player performs a positive combination. If the element is not adjacent, then it is a negative combination. If the hero attacks with the same element that is already applied, then a neutral combination is performed (Figure 12).



Figure 12 – Demonstration of how elemental combinations work on our game. In this example, if the attacker (Fire) hits fire, it will generate a neutral combination. If it hits the adjacent elements to fire (earth or wind) will generate a positive combination. If it hits any non-adjacent element (lightning or water) then it will produce a negative combination.

3.6 Game Requirements

As our prototype, the game would require several systems: an ability system, an element power system, a combinations system, heroes, teams, objectives (towers), a map, networking and a database. There must also be a system responsible for the game loop and to check the winning condition.

There are two teams in the game, each having three heroes. The team must have a name, maximum and current resources. Every first, second and fourth ability a hero unlocks, its team gains one more resource to use. Each team has two towers they must defend. The game ends when a team destroys all the other's objectives.

Each hero in the game must have information about its characteristics: health, polarities restriction, abilities available to unlock, unlocked abilities, damage to objectives and its element. It must also have a list of effects currently active on it, information if it is alive or dead and if it has played or not. It must also have information if it has attacked, moved or damaged an objective.

The game loop consists of 2 main phases: turns and rounds. A turn begins when a hero starts to play and ends when that hero has no more actions to perform or decides to end its turn. During its turn, the hero can move, attack or use element powers. When the turn ends, the hero is marked as having played already and cannot play again until the next round starts. A round encapsulates one turn for each player. After the hero play, the combination system kicks in and checks if there is any combination to solve this turn. After dealing with the combinations, the turns end and the other team starts to play and choose a hero to start its turn. This process repeats until all heroes have played. Once every hero has played, the current round ends and a new one begins. At the end of the round, all abilities on cooldown are restored to their default condition and the teams regenerate all resources spent. Any active element power in the element power zone is discarded and each team can activate a new one if they have one available.

The ability system will be responsible for having classes for holding the abilities' data and classes for processing the abilities' mechanics. There is a total of 48 abilities, 8 for each hero. Each ability always has a target (hero, tile), a maximum and minimum range, a cost and a polarity. Every time an ability hits a target, the combination system calculates that combination and updates the stored information about how many positive, negative and neutral combinations the hero has. As mentioned previously, at the end of each turn, if there are any combinations to solve, i.e., there is at least one hero with at least one positive, negative or neutral combination, the combination system starts solving the combinations. If the hero has more positives than negatives combinations, his team gains an elemental power from the same element as him. If it has more negatives, then the opposing team receives the element power. If the number of positives is equal to the number of negatives, then both teams gain a power.

Like the ability system, the element power system also has classes responsible for holding information for each power and classes responsible for processing the logic behind each power. There is a total of 55 different element powers, 11 for each element. There are three kinds of powers: instant powers, the user activates them as soon as they gain them, store for later powers, where the user can save the power and use when one of its heroes is playing, and element zone powers, that can be used at the beginning of each round and persists until the end of that round.

A hexagon tile map is required for players to navigate. This map must have several types of tiles, like starting and respawns tiles, objective tiles and normal tiles. Each tile can only be occupied by one hero or objective, but every tile can be used for movement, i.e., the player can use an occupied tile for its path, providing he does not finish his movement on that tile.

The database must contain all information pertinent to the user account, both their personal information, such as email or username, as well as all information regarding the game, such as all heroes unlocked and their levels. In addition to the user information, the database must also hold all the information about the game itself, as it would be used for a data driven design to facilitate the balancing process once the time arrives. The database must also keep a log of every action taken in every game, to allow the reproduction of the same game later.

Finally, there must be a networking system which keeps both players synchronized about the game state. The dependence to this system must be minimal, since at a later stage it will be replaced by a remote authoritative server.

Table 1 – Table with full description of functional and non-functional requirements

##	System requirements
Functional requirements	
1	The player shall be able to create an account.
2	The player shall be able to login to the game.
3	The player shall be able to check its profile information.
4	The player shall be able to buy new heroes in the game.
5	The player shall be able to see each hero's abilities.
6	The player shall be able to see each hero's information.
7	The player shall be able to create new hero pre-sets.
8	The player shall be able to see each hero's pre-sets.
9	The player shall be able to activate a pre-set.
10	The player shall be able to edit a pre-set.
11	The player shall be able to choose the team they want to play with.
12	The player shall be able to check its current account currency.
13	The player shall be able to check its account statistics.
14	The player shall be able to change the game's video settings at any time.
15	The player shall be able to change the game's sound settings at any time.
16	The player shall be able to exit the game at any time.
17	The game shall have a turn system.
18	The game shall have a round system.
19	The game shall implement all Heroes existing in the board game
20	The game shall implement all Abilities existing in the board game
21	The game shall implement all Element Powers existing in the board game
22	The game shall be able to hold information about the state of the Heroes.
23	The game shall be able to hold information about the state of the Abilities.
24	The game shall be able to hold information about the state of the Element Powers.
25	The game shall have a deck of Element Powers for each element in the game.
26	The game shall be able to hold information about the state of the Tiles.
27	The game shall be able to hold information about the state of the Map.
28	The player shall be able to see both team's resources at any time.
29	The player shall be able to see both team's towers health information at any time.
30	The player shall be able to see both teams scores at any time.
31	The player shall be able to know at any team which team is playing.
32	The player shall be able to know at any time which team belongs to them.
33	The player shall be able to see at any time the last five actions in the game.
34	The player shall be able to see at any time the game's duration time.
35	The player shall be able to see at any time both team's heroes game status.
36	The player shall be able to see at any time both team's hero's abilities bar.
37	The player shall be able to see both team's hero's element combinations at any time.
38	The player shall be able to see both team's hero's health at any time.

39	The player shall be able to see both team's element zone element powers activated at any time.
40	The player shall be able to use an element zone power at the beginning of the round.
41	The player shall be able to see both team's number of element powers stored at any time.
42	The player shall be able to know which hero is currently playing at any time.
43	The player shall be able to see each owned element power stored.
44	The player shall be able to see its hero's turn actions at any time.
45	The player shall be able to see more information about their hero's ability.
46	The player shall be able to see more information about their owned element powers.
47	The player shall be able to use a hero's ability.
48	The player shall be able to use an owned element power.
49	The player shall be able to attack a tower.
50	The player shall be able to unlock new abilities.
51	The player shall be able to fill ability's hero targets.
52	The player shall be able to fill ability's tile targets.
53	The player shall be able to fill element power's hero targets.
54	The player shall be able to fill element power's tower targets.
55	The player shall be able to fill element power's tile targets.
56	The player shall be able to see a timer to use a counter ability.
57	The player shall be able to see a timer to activate an element.
58	The player shall be able to see a turn timer.
59	The player shall be able to resurrect a hero by choosing an available tile in the map.
60	The player shall be able to spawn a hero by choosing an available tile in the map.
61	The player shall be able to move a hero by choosing an available tile in the map.
62	The player shall be able to see the available movements in the map.
63	The game shall be able to determine which team won the game.
64	The player shall be able to determine which team won the game.
Non-functional requirements	
63	The game shall reutilize as much code as possible.
64	The game shall be able to run in older hardware computers.
65	The game shall be able to support easy modifications to its structures.
66	The game shall be able to support addition of new content without modifying old content.
67	The game mechanics shall be able to fully integrate with the GUI.
68	The game shall be able to modify the balancing values through the database.
69	The game shall support multiplayer through the network.
70	The game shall be independent of the networking system.
71	The game shall communicate with the networking system through an abstraction.

PART II: IMPLEMENTATION

4 Development process

In this section, we will provide a brief description of the development process for the whole project. Each of the following paragraphs has a short description of what was done during that development cycle. These steps are described further in the implementation section.

Given the nature of the project and the requirement being quite fixed, we decided it would be better to follow a Waterfall development model [35] instead of an Agile approach [36]. This way, we tested each system extensively before moving on to the next, reducing the number of bugs when combining all systems to make the final product. We started by planning the work division and defining the responsibilities of the DAP and the MTP. We created a GANTT chart, with the tasks division, critical tasks and milestones, which were points where both projects should be combined to realize a prototype test. The full version of this GANTT can be found in appendix 9.1.

We then proceeded to define the database for the game. As mentioned before, all information regarding the balance of the game, such as the heroes' health, abilities and power should be present in the database in order to make balancing easier. This would allow us to tweak this values and test them without the need to recompile the game. The database should also have all information for the user's account such as username, password, content owned and profile. The final part of the database should contain information that allowed us to store a history of the game. In this history, there should be enough information to completely recreate a match.

Once the database was defined and tested, we proceeded to define which classes would be required to store in the game the information existing in the database. These would be the core information classes for the game, such as abilities, EP or heroes. At this point, parallel work started with DAP [1]: they would work on the GUI to display all the information required for the player while we would work on all the queries and functions needed to provide information from the database. This part of the work consisted on developing and testing the login system and the main menu for the game. In the menus scenes, we should provide the GUI with information about every hero and ability in the game as well as, allow the user to choose which heroes to control in a match and allow them to buy heroes they do not have unlocked yet. A big portion of time was dedicated to this task, since many of the structures developed at that time would be reutilized later for the in-game scene. The most relevant system developed at that point was the abstraction that allowed us to easily change between different database implementations and the encapsulation of the queried information from the database.

A critical component of this project was the networking: since the game was multiplayer, in case of a failure in this task, the game would have to be reworked conceptually to work offline. Several technologies were considered for this part, including Photon Unity Networking (PUN) [37] and native U3D networking (UNet) [38]. Since for this prototype we did not require a dedicated server, we decided to use UNet for now and work with abstractions to facilitate a conversion to dedicated server later. A simple prototype was created where two clients would connect and see each other's movement through the network. This allowed us to get a grasp of how the networking worked and an insight on how we would build the systems for our game. A tutorial from U3D was used for this [39]. It is worth mentioning that, although this prototype did not require an authoritative server, we implemented an abstraction that allowed us to replace all the network later with minimal effort. This was done because we wanted to have an authoritative server for the game but, that was beyond the scope of this project.

We then proceeded with the development of the tile map for the game. This was a simple task, since the map can be considered as being 2D, having only x and y coordinates. Some considerations had to be kept in mind while doing this, since our map uses tiles instead of squares. To assist us, we used several algorithms developed by Red Blob Games, available publicly on their blog, where they compile more than 20 years of experience in developing tile map algorithms [40]. We ended up using two types of coordinate systems: an Offset coordinate system which is used for storage and an Axial coordinate system to use on all our algorithms. Conversion from one system to another is very simple and it is explained in chapter 4.2.3.1.

To make the creation of the map simpler, a map editor was also developed. This editor works inside the U3D editor and allows us to set several parameters for the map and which tiles can be used for movement or not (Figure 13).

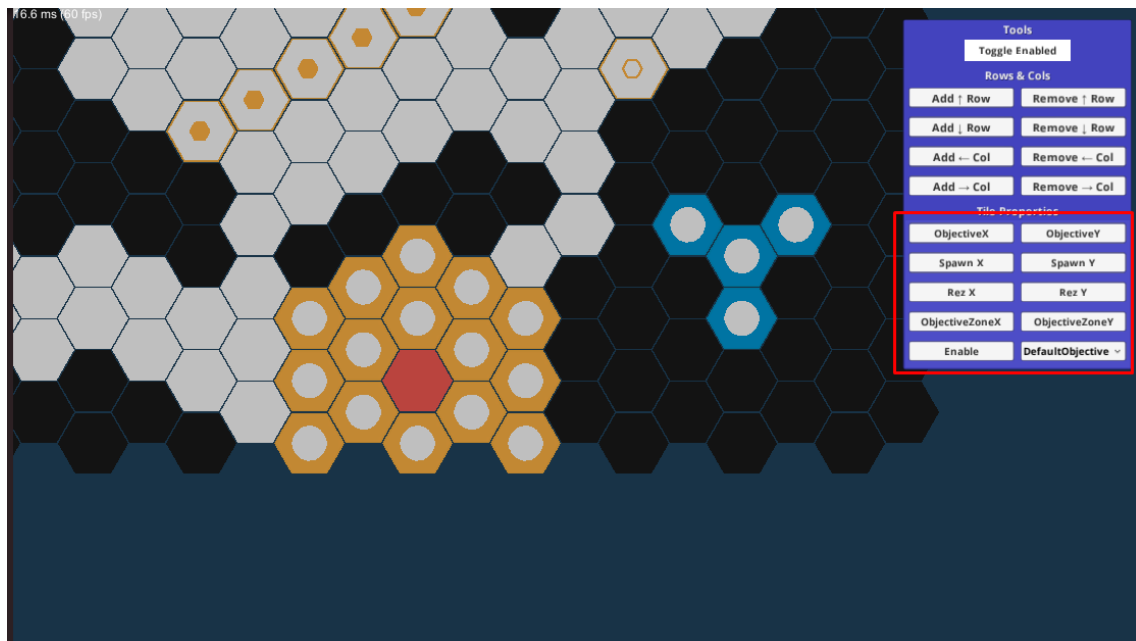


Figure 13 – Map Editor developed for us to create our map. The panel on the top right of the screen displays the properties of the selected tiles (blue tiles). It is possible to change these properties by clicking on the highlighted buttons.

After the map was created, it was time to start integrating all the developed systems so far in a single project. In this build, players could login, choose their team, enter the map and see each other. The game mechanics were not implemented yet.

Once the map, networking and database were all integrated, it was time to start implementing the specifics of the game mechanics: the heroes, abilities, EPs and game rules. The first of these systems to be implemented was the movement system, which used an A* algorithm to find the path between the hero and the selected tile. After the movement, we implemented the Ability system and all abilities of one hero. This was done in order to test the system before advancing to the next one. Then, it was time for the EP system and, as with the abilities, we started by implementing only one element to see if it worked. Once both systems were performing as desired, we implemented all the game rules, such as movement limitations, the turn and round systems and the winning condition. When all systems were implemented, we went back and implemented the rest of the heroes, abilities and element powers. These systems represent the major part of the development cycle of this project.

Once all the systems were in place, we tested our theory that we could easily transfer the game from the UNet system to our own server. We developed a Node.js server and, using the abstractions in place, we ported the game with little effort in a very short period (two days).

During the development of the project, we used subversion software to keep the code synchronized and to keep all versions of the code. The software used was Microsoft's Team Foundation Server [41], which can be natively integrated with Visual Studio for ease of use [42]. This kind of software allows for parallel coding and offers tools to help merge the code, which is usually done automatically by the program.

4.1 Game Technologies

Although many technologies are used to develop a game, in this section we are going to focus on Game Engines, Database Management Systems (DBMS) and networking in games.

4.1.1 Game Engines

Game engine is a computer program, or set of programming libraries, used to simplify the process of developing digital games. A common set of features of these programs are a graphical engine, used to render 3D and 2D graphics, a physics engine, used to simulate physics and collisions and support to animations, sound, artificial intelligence, networking, memory management and a scripting language [43], [44].

While the first game engines were built in lower programming languages, some even in assembly, newer engines are being built in higher level languages, such as Java and C#/.NET, Python or Lua Scripts. With the higher capacity of today's CPUs, the bottleneck of performance for games lies on the graphics card, making the potential slowdown due to translation overheads of higher level language negligible, while increasing the productivity gains of these languages [45].

For this project, we decided to investigate three of the most popular game engines: Unreal Engine 4 (UE4) [46], CryEngine V (CE5) [47] and Unity3D 5 (U3D) [48]. Many features are important to consider when choosing a game engine: the graphical fidelity, which shaders are available, how "real" is the physics engine, how optimized the engine is and if you have access to the engine source code, to allow you to shape the engine to your needs and requirements. For us, the most important features were the ease of use, our knowledge of the used programming language and the price.

U3D has a large and active community. Plugins and add-ons can be found online on the asset store and community sites [49]. The community forum is also full of professionals who frequently answer questions and exchange snippets of code. UE4 and CE5 also have asset stores and communities' forums, but they are not as active as U3D's and they are much more expensive.

UE4 has improved its programming language in the current version, changing it from UnrealScript, a proprietary programming language from Epic Games, to C++, a more commonly known and less convoluted language [50]. Although CE5 has support for C#, at the time it only offered C++. This was another disadvantage comparing to U3D, which uses C#, a higher-level language than C++, which allows developers to enter the game development process easily. One of the greatest advantages of C# is that it is a managed language, meaning that memory allocation, covering memory leaks and other similar process are done automatically by the language [51].

Another big advantage of U3D is how simple its user interface is. U3D hides all unnecessary information from the user and leaves only the essential visible. It also removes all the functionality from objects and the game designer can add components (scripts) to give only the functionality required for it to work. To add more components, the user can add pre-existing scripts or create new ones, by creating a C# class which extends MonoBehaviour. This means that the user interface remains much cleaner and less cluttered with information (Figure 14).

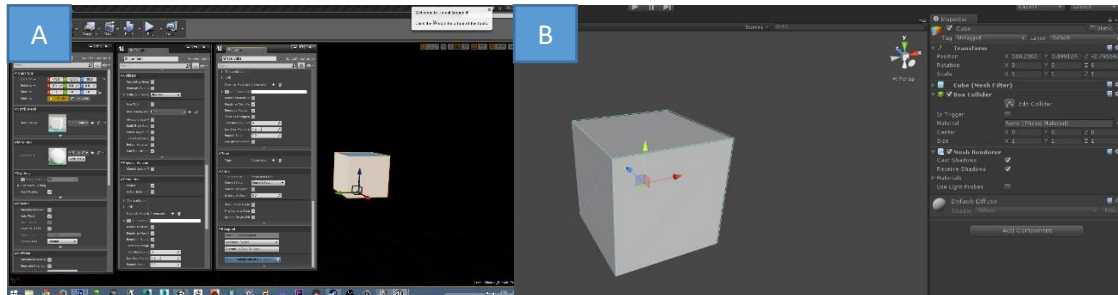





Figure 14 – A: UE4 displaying all the properties of a cube at creation time. B: U3D showing all existing properties of a cube at creation time.

Although the other game engines analysed for this project are overall better than U3D, two factors were decisive when choosing it as our game engine: it is by far the easiest to use and, at the time, it was the best free game engine. In Table 2 there is a summary result of the comparison.

Table 2 – Game Engines features comparison

	 unity	 UNREAL ENGINE	 CRYENGINE
Programming Language	C#, UnityScript and Boo	C++	C++
Networking – Dedicated server	Not available	Available	Available
Networking – Peer to Peer	Available	Available	Available
Source Code Access	No	Yes	Yes
DirectX Support	9, 10 and 11	9, 10 and 11	9, 10 and 11
Multithread support	Not available	Available	Available
Price	Free until 100.000\$ annual revenue [52]	Free with 5% royalties after the first 3.000\$ per quarter [53]	50€ per user for basic, 150€ per user for premium [54]

4.1.2 Database analysis

We also needed some way to store information about our game. We wanted to use a data-driven approach while developing the game so, we could easily update and test the game's balance. The most common way of doing this, in gaming development, is the usage of .ini files with a simple 'Tag=Value' syntax [55]. This would be sufficient for testing and development, but since our game will be online and will have a competitive multiplayer component, we cannot let the player have access to the balancing files. Considering we also needed to store information about the player, such as username, password and billing information, we then decided it would be best to use an SQL database.

As for the system, we considered NoSQL database and traditional relational database (RDBMS). NoSQL has gained a lot of traction in the past years, but there are still a lot of limitations with this technology. The most noticeable is the fact that complex querying is not supported and there is no way to use analytical tools to evaluate the database [56] (Table 3). This, combined with the fact that we had previous experience working with RDBMS, MySQL specifically, and we already knew how to use C# to communicate with the database, led us to choose MySQL for our database.

Table 3 – NoSQL databases pros and cons compared with RDBMS

NoSQL	
Pros	Cons
High scalability	No standardization
Distributed computing	Limited query capabilities
Lower cost	Eventual consistent model is not intuitive to program for
Schema flexibility, semi-structured data	Unpredictable data

We used WampServer [57] to host our database, MySQL Workbench [58] to design it and MySQL Connector [59] to query the database from C#. Please refer to Architecture chapter for more details on how we did this.

4.1.3 Networking

As our game is a multiplayer game, there must be a system responsible to connecting and synchronizing both clients. There are several solutions for this kind of technologies, being the most common peer-to-peer and client-server networking.

Peer-to-peer connection is the simplest of the two: it just connects two players and one of them will act as a server (this is called a host). The advantage of this approach is that reduces the overhead on the server, since its only job is to connect two peers and allows the same code to be used on both ends. The downside is that latency in one of the clients can be felt by every player connected to the same session and it is very easy to cheat, since there is no server side validation of player actions [60].

The client-server approach, on the other hand, gives total control to the server. It is responsible for relaying all information from client to client and, by doing this, it can validate all the actions of the players: if it detects something wrong, it can answer to the player by denying an action. As it is evident, the overhead on the server is huge, since it must validate all the actions for all the players in the game and then relay the information to the relevant clients. The server must also retain an abstraction of game states from every match being played. This is the price for an authoritative server. On the other hand, it is possible to use client side prediction to give the player a smooth gaming experience, avoiding to a certain degree, the effect latency has on the games [61].

For our game, we started with a peer to peer connection for the proof of concept prototype and later, advanced to a client-server implementation. The peer-to-peer implementation was made using UNet [38], U3D's networking, and for the client-server implementation we developed a Node.js server [62].

U3D provides a high-level API for developers to facilitate the implementation of networking on their games. It is built directly on top of the transport layer of the low-level API and handles many common tasks required for networked games (Figure 15). Aspects of networking, such as message handlers, general purpose high performance serialization, distributed object management, state synchronization and network classes are all handled by the high-level API. Although many topologies are supported by the low-level API, the high-level only supports an authoritative server topology, with one of the clients doing the part of a client and server at the same time (the host).

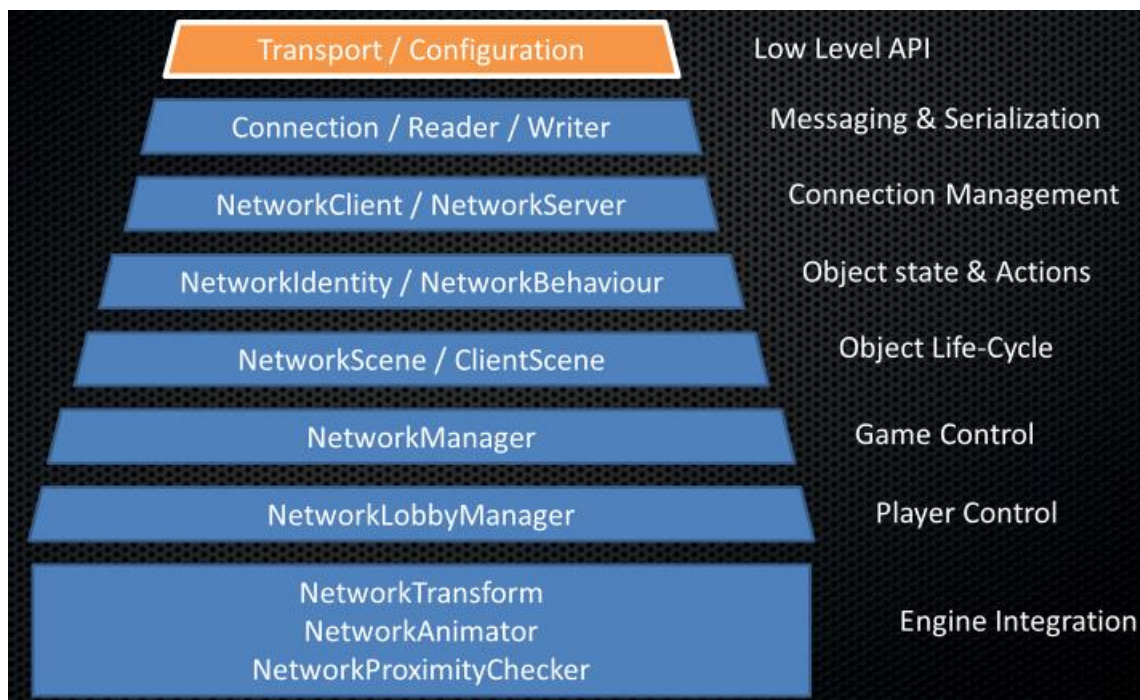


Figure 15 – U3D’s High-level API for network games. Uses their own low-level API to handle the most common requirements for multiplayer games [63].

As mentioned before, in U3D’s networking, one of the clients must also do the part of a server. Using the U3D’s internet services, it is possible to deploy a multiplayer game with little effort. U3D provides a lobby network class which allows the developers to quickly setup a matchmaking logic and connect two or more players in rooms for the game. Each room works as a separated instance of the game in the server, each one with its own set of variables.

Another solution UNet provides, is to have variables that are automatically synchronized in all clients. To do this, a script must inherit from NetworkBehaviour (instead of MonoBehaviour) and have the attribute SyncVar before the declaration of a variable field. Only primitive types and U3D’s types, such as Vector3, can be marked with SyncVar. Although this feature would help in the prototyping phase, we purposely avoided this solution to minimize the amount of change required in the code once we removed UNet from the project and replaced it with our networking code.

One key function of an authoritative server implementation is the ability to validate the user input before answering to it. U3D’s allows this by the usage of remote actions: a client wants to perform an action and sends a request to the server (a Command). The server processes the request and answers accordingly to the sender or all connected clients (a Remote Procedure Call – RPC) (Figure 16).

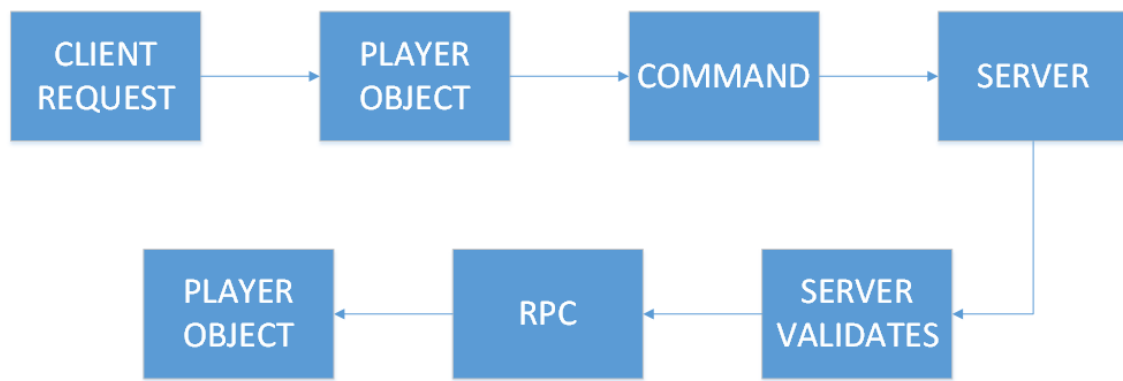


Figure 16 – Client-Server communication pipeline in u3D. Every action the client want to perform is sent to the server via a command. The server process the data and send information back to the client through a RPC [50].

Node.js is a server-side platform, developed by Rayan Dahl, which is built on Google Chrome's engine (V8 Engine). This platform is very fast, scalable, easy to use and it has become extremely popular in 2016, with major companies, like Paypal and E-bay using this technology for their server side applications [64]–[66]. There are several reasons for this rise in popularity, but it is mainly due the fact that it has impressive benchmark results, a low entry barrier for developers and, a huge and active community willing to help and share code online [67].

There are also several online courses explaining how to set it up to use with U3D. One of these courses is Alexander Zanzfir Pluralsight's course 'Unity Multiplayer Game Development' [68]. Following the instructions on this course, using Node.js for multiplayer in U3D is extremely simple. We start by adding Socket.IO support for U3D, with an asset available for free in Unity's asset store [69]. Then we install Socket.IO in Node.js using npm [70], which is a JavaScript library that allows network communication between clients and servers via sockets. The communication is done by sending and receiving packages with JSON objects. Using this implementation means that all communications must pass through a game object containing the SocketIO component, which means it works in a very similar way of UNet. By taking some precautions with the code, switching between UNet and the Node.js implementation only affects one class in the client side. Whoever, the server side implementation must be remade in JavaScript.

In 2015, the new ECMAScript (ES6) reference used by JavaScript was released, which was a huge leap for the language [71]. This change introduced several new concepts and 'sugar code' that eased the development of code for JavaScript, making it more similar to other high level languages, such as C# or Java. Most relevant to note is the introduction of classes and array helper functions, that work similarly to LINQ queries in C# [72]. All of this, made Node.js even more appealing, since currently, 97% of ES6 features are compatible with Node.js on the server side [73].

We also looked at other solutions, namely PUN [37] and our own implementation of web sockets. Our implementation of web sockets would be the closest to what we wanted to achieve in the end, i.e., an authoritative server, but at the time U3D had lots of issues with web sockets and we were not able to bypass these difficulties [74]. As for PUN, it was a solid consideration and we started to implement using this technology, but there were several factors that led us to go back to UNet: PUN is lacking in its documentation, it does not have a support system for questions and answers such as U3D has and, payment is required to use this technology with more users [75].

4.2 Implementation

In this section, we will explore in more detail how the code was organized and how we implemented all the required systems. We will start by showing how the systems were planned and then how we used these systems to implement all the game mechanics existing in the board game. Since we were following the Waterfall Model for software development, each system would be fully implemented and tested before advancing to the next one, minimizing the number of bugs produced by adding more systems to the game.

4.2.1 Architecture

Following the work division explained in the Introduction chapter, we had to organize the project in such way that we could work in parallel with minimal dependencies from each other. Also, as mentioned before, DAP's team was responsible for the Aesthetics and History, so they had the responsibility of developing all the aspects of the GUI, 3D, animations and particles, from concept to deployment. Meanwhile we were responsible for the Mechanics and Technology, so we had to develop all the backend supporting systems, such as the ability and EP systems, map builder, game logic and networking. We were also responsible for using the developed systems to implement the mechanics of the game.

4.2.1.1 Database

To develop the database, we started by looking at the requirements (see section 3.6). We defined every relationship required in our database and then, we organized them in three logical parts: account, game data and logging. After the entity relationship diagram was created, we started to add attributes to the entities. We then performed extensive tests to the database, to ensure that all the information required was present. Once we were satisfied with the result, we started implementing it on the MySQL Workbench.

For the user account there is a central entity, named account. It has relationships with the profile information, which contains the amount of gold, experience and avatars the player owns. It also has information about all purchases the player made, thus every content the player has unlocked, such as heroes, weapons, enchants and skins. There are also several other relationships used to display information about the player inside the game, such as Achievements, Statistics and Titles. Lastly, the pre-set relationship holds information about which pre-sets the player has configured and which ones are active (Figure 17).

The game data, used for balancing the game, has two central entities: heroes and EPs. The hero has information about itself, i.e., health, difficulty, role, how much damage it does to objectives and which polarities are available for each ability slot (Figure 18). There is also information about how much in game currency the hero costs and its codename, that is used by DAP to display information on the GUI [1]. Other balancing information is about the objectives, how much health it has and how much damage it does.

The history section of the database was extensively tested. During weeks, we thought about every information required to be able to reproduce an entire match: which hero the user played at turn X, which ability was used, which were the combinations generated that turn and the element resolved. How many times a hero died during a match, how long was the match and who won, among many others. The paper prototype was crucial at this point, since it was much easier to look at the board game and think what information we needed, instead of just trying to do abstract thinking. We created 42 entities and numerous relationships between them to record all information needed for the history. Although this section of the database can handle

all the information required, due to the time constraints, we decided that we would not implement it in this project. It was, however, developed and deployed with our database, just with empty tables and no queries performed to it.

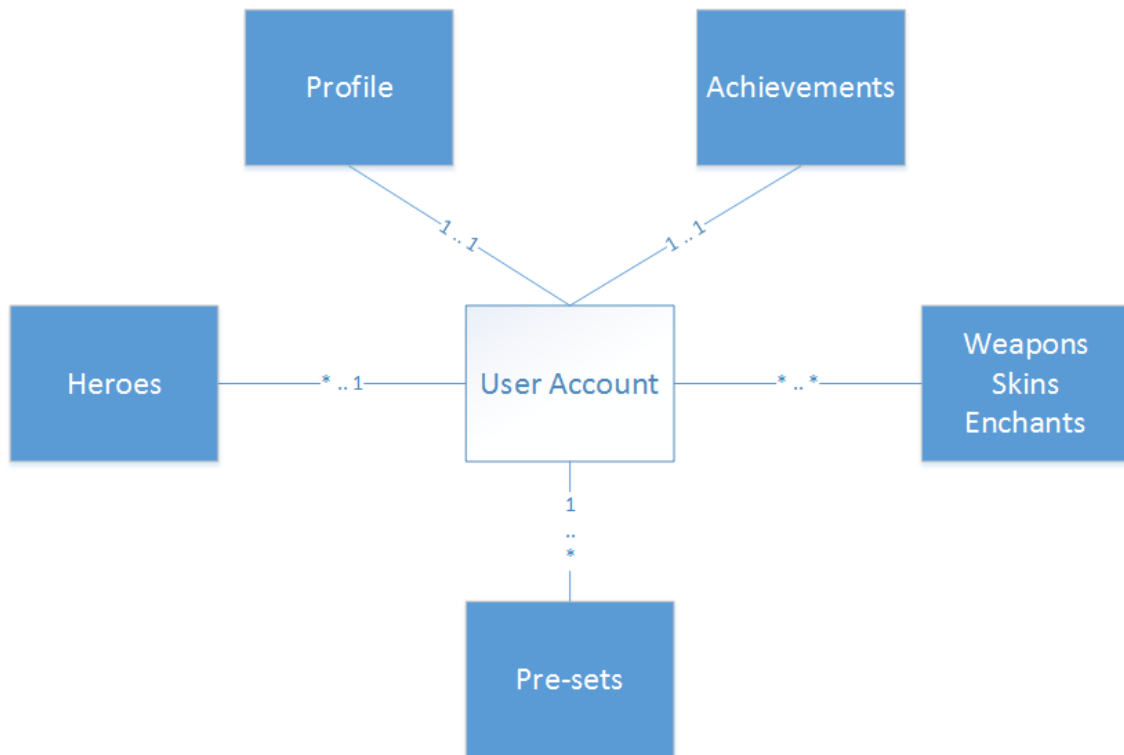


Figure 17 – Relationships between the user account and their possessions and achievements in the database. Full version can be found in appendix 9.4.1.

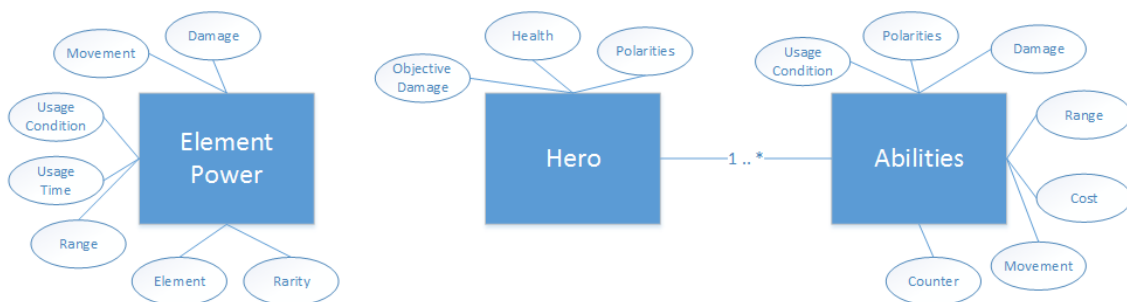


Figure 18 - EP, Hero and Abilities entities in the database and their relationship. Full version of this section of the database can be found in appendix 9.4.2 and 9.4.3.

4.2.1.2 Data structures

A project of this scope requires a lot of planning and structuring to avoid complications later. We started to define which structures sections were needed: a local cache of the database, information that will be only used to display in the GUI, information that belongs to each user and information that will be used for the game logic calculations. We also needed a set of classes to create, edit and load the map. Classes for querying the database and to handle the networking were also required. In Figure 22, each blue area refers to a section of data structures.

Since the DAP was already implementing the GUI before we started the development of the data structures, they defined all the classes they needed to store information for the GUI. Later, after we finished creating the database, we used these classes and updated them to store all the information required. These classes, which we called the Core classes, have all the static data

about a certain element in the game. For instance, the HeroCore class has information about the hero's in-game stats, their name, value, role, difficulty, available abilities and skins (Figure 19). The core classes contain data that does not change over time.

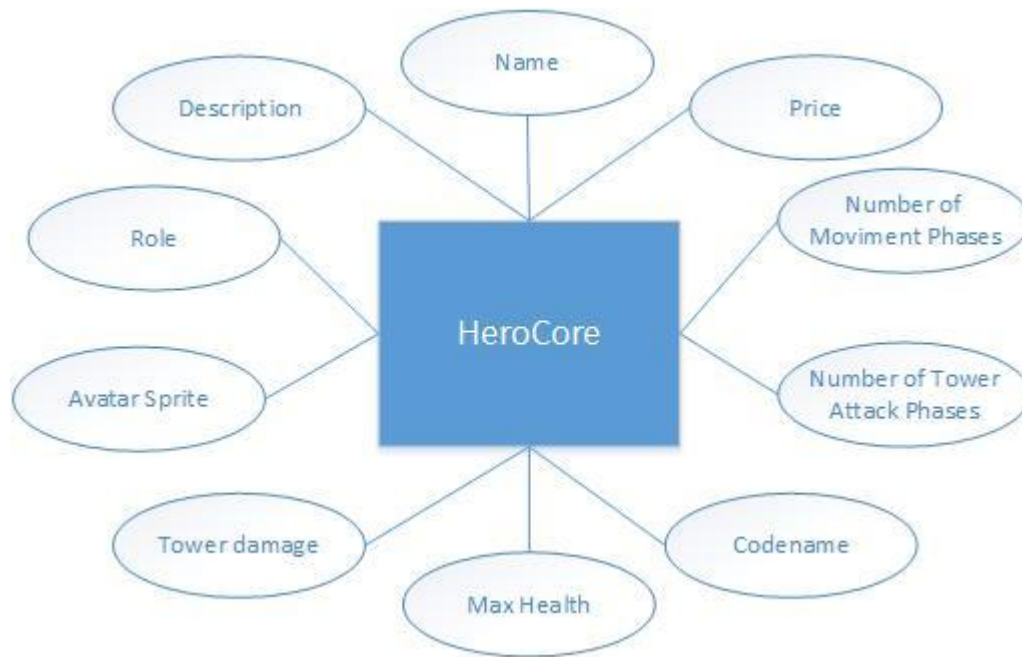


Figure 19 – HeroCore class and all its properties

The core classes were then encapsulated by the concrete classes, the classes that have data about a specific instance of a hero, ability or element power, which we called Ingame classes. At this point, a mistake was made while developing this encapsulation. Originally, we had inverted the encapsulation and the IngameHero was encapsulated by the HeroCore. This meant that, when instantiated, there would only be one IngameHero per match and several instances of HeroCore. A consequence of this was that if two teams had the same hero, once one hero used an ability, this ability would be on cool down for both teams. As soon as we detected this problem we inverted the encapsulation and removed redundant classes. In the end, the IngameHero has a pointer to the HeroCore, this way gaining access to all the static information, and has a set of data specific to its instance: which are their chosen abilities, how much health it currently has, if it has played this round, if it is playing and which is its unique identifying key, among others.

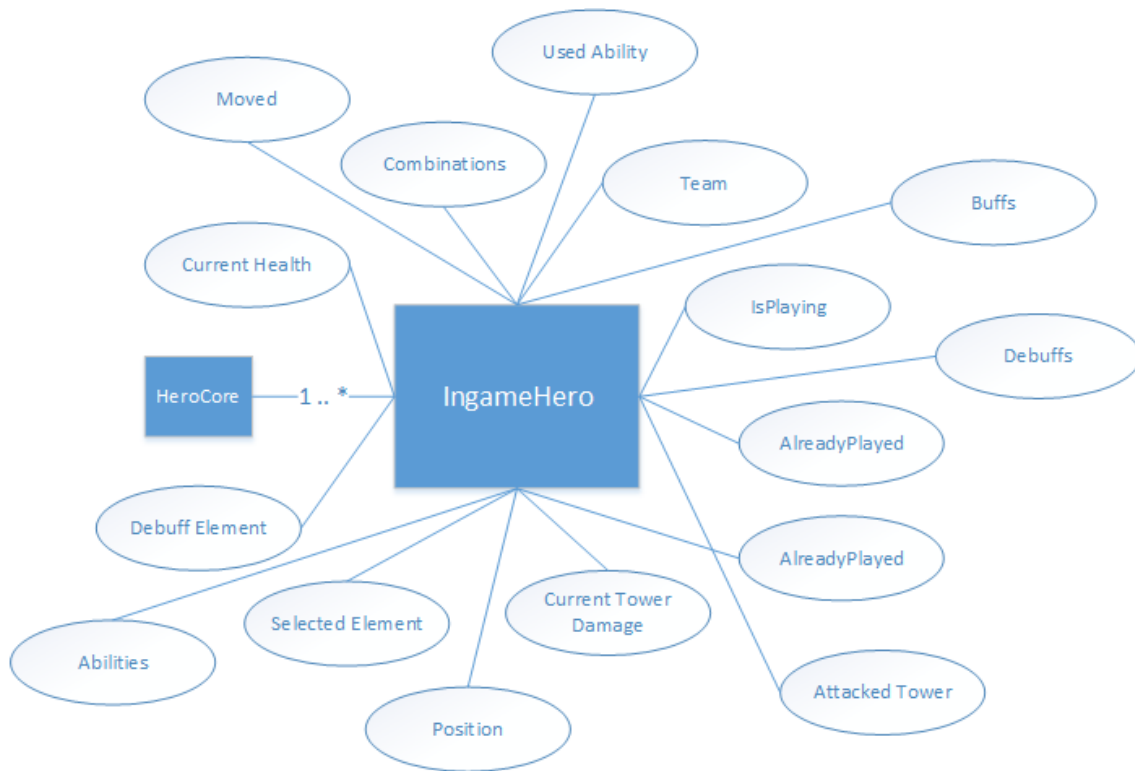


Figure 20 – Ingame hero with all its properties and a reference to the HeroCore.

This strategy was used for heroes, elements, EPs and abilities. Sometimes more levels were needed. Looking again at the hero as an example, there is another layer of instantiation, the hero that belongs to a specific player. This hero has an active set of skins, weapons and enchants, an amount of earned experience, a level and a set of statistics, such as number of times played and most used ability (Figure 21).

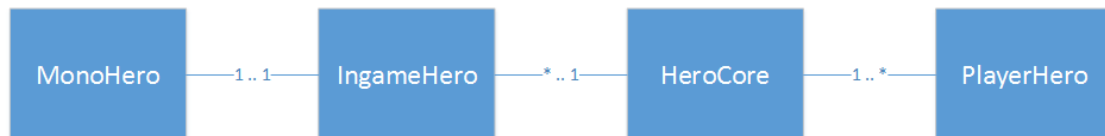


Figure 21 – Complete hero hierarchy and the relations each type of hero has with other heroes.

As mentioned before, there were also classes responsible for querying and caching a copy of the database to a local database. This was done by querying the database and then storing the information in the local cache. For the querying, we used the abstraction mentioned before, where we had three interfaces, each with its own responsibilities in the database: one for getting information from the database, one to insert or update data and one to validate information given by the user, such as the login. These interfaces were IDbLoader, IDbUpdater and IDbValidator, respectively. All the queries were processed by a DbHandler class, which was responsible for translating them to SQL and to get the result back. The information was then stored in the LocalDb class, where there are several dictionaries with all the balancing information. All the dictionaries use a string as a key and an object of a specific class as a value. For instance, for the HeroCore dictionary, we have a key value pair of HeroCodename – HeroCoreObject.

There are two important advantages of using this local cache. First, since the game information does not exist locally, it is much harder for the player to cheat or to hack the game by changing

values in the memory. Secondly, since the cache is loaded every time the game initializes, every time the user opens the game all the values of his local copy are synced with the most recent values from the database, without the need of any code compilations and deployment.

By using the abstraction described above, we ensured that we could easily expand the game in the future if we wanted. To add a new hero, we just need to add its values to the database, without coding anything. To add a new ability, we just extend the `IngameAbility` abstract class and override any method where we need to implement exceptions. The same process can also be performed with Elements and Element Powers.

4.2.1.3 Patterns used

To allow the parallel implementation of the work, we used two techniques: we followed the MVC design pattern (Figure 22) and we developed an abstraction to allow the simulation of the database so DAP's team could start the implementation of the GUI before the database handling classes were implemented (Figure 23).

The MVC pattern allowed us to develop the Views and the Models in parallel: while the DAP was designing and implementing the GUI, we were defining the data structures and querying the database. The controllers were also implemented by the DAP team while they were developing the GUI, using the database abstractions to have access to dummy information to display.

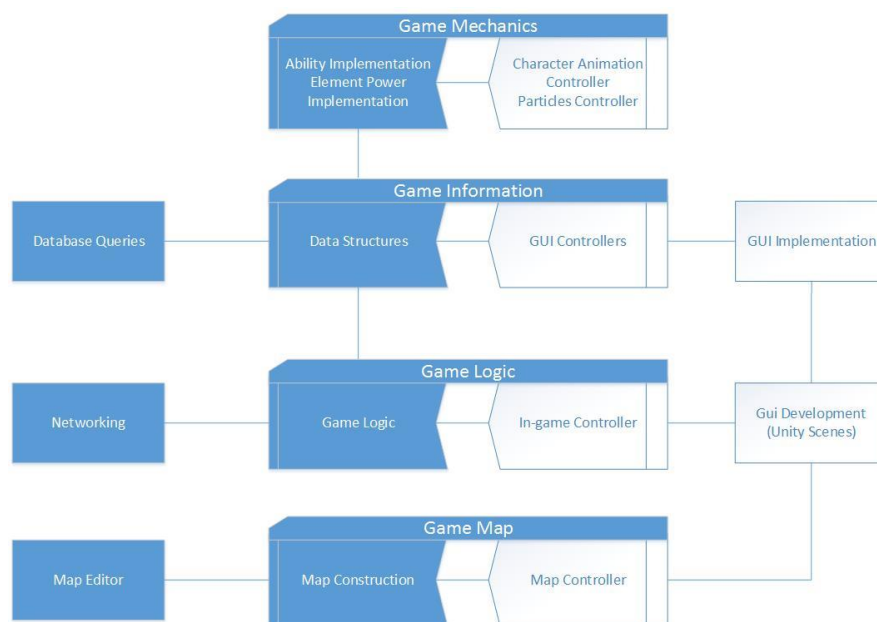


Figure 22 – Division for MVC implementation. On the left, there are the Models. On the right, there are the Views and in the middle the Controllers.

For the database integration, we used a local database class (`LocalDb.cs`) to cache a copy of the balancing information from the database. The `LocalDb` used the abstractions to communicate with the database and to populate the cache. This cache, either populated with real information or with dummy information, was then used in DAP to populate the GUI. Since they were only dependent of the local cache of the database, once the implementation of all the data structures and queries was complete, there was no need to change anything in the GUI.

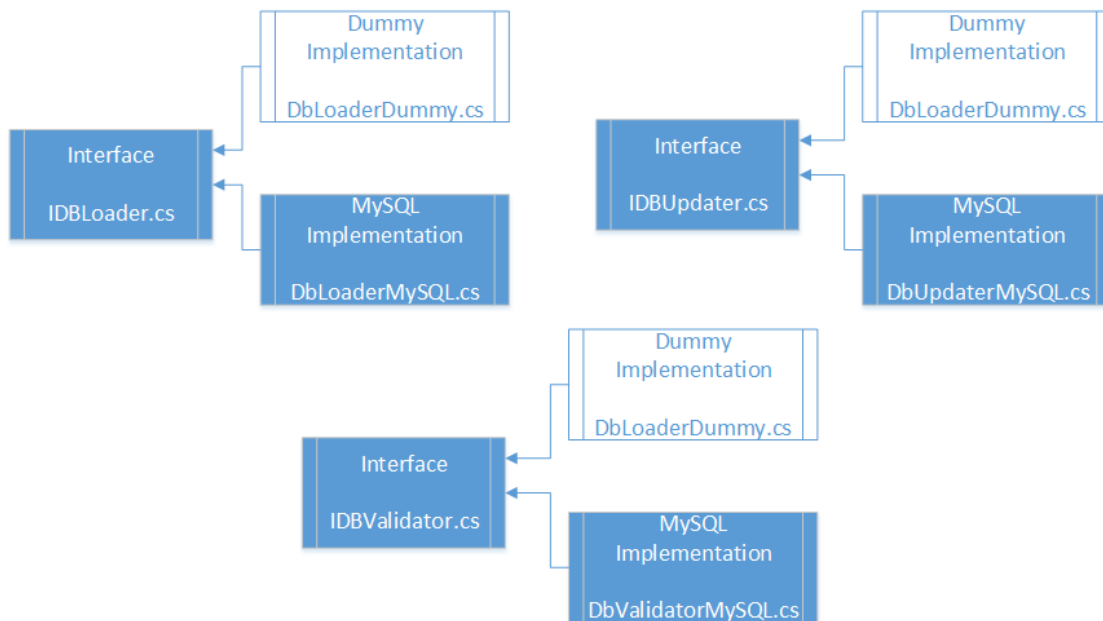


Figure 23 – Implementations of the database abstraction. Blue represents what was developed in this project, white was implemented in the other project

As it is usual when using this kind of abstractions, a factory had to be implemented. This factory looked at an Enum (set in U3D) to instantiate the proper database handler class (Figure 24). A factory was also used to instantiate Abilities and Element Powers.

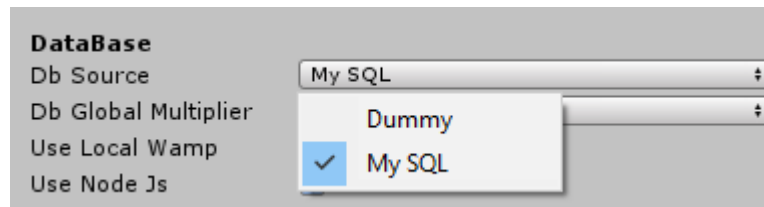


Figure 24 – Enum set in u3D Editor to choose which database handler to use by the factory.

Other patterns used were adapter, facade and singleton [76]. Adapter was used in order to allow one of our structures (a MonoHex, the structure used to represent a tile) to be placed inside a priority queue developed by Blue Raja [77]. Facade was used throughout the code, wherever there was a repetition of function calls, to help with maintainability of the code. Finally, singleton was used when we wanted global access to a U3D script. The singleton implementation in U3D differs from the traditional implementation and this gave us a lot of problems. After a lot of research, we found that the implementation consists of two lines of code in U3D (Figure 25).

```

public class SingletonExample : MonoBehaviour
{
    internal static SingletonExample Instance;

    void Awake()
    {
        Instance = this;
    }
}

```

Figure 25 – Code snippet for the singleton implementation in U3D

4.2.2 Networking

To use U3D's networking (UNet), the code must be structured in a specific way. All outbound communication must be made from a single object, called the player object, and all communication from the server is collected by that same object. It serves as a router for all communications. The advantage of this is that, since all the network code must be gathered in one single object, when it is time to change the implementation, that's the only file that needs to be changed.

Another particularity of UNet is that both the server and the client share the same code, meaning that there is no need to code two versions of the game. If a function is supposed to be executed only on the server side, then the Command attribute is added to it (Figure 26). When the server wants to communicate with the clients, it must evoke a Remote Procedure Call (RPC). This function is evoked by the server and executed by all connected clients. It has a similar signature as the Command, it must have an attribute ClientRpc on it and receive the same types of parameters as the Command.

```

[Command]
public void CmdExample(/* parameters must be primitive types or arrays of it
*/)
{
    /*
     * code executed only by the server
     */
}

```

Figure 26 – Code snippet for a Command. This code is executed only on the server side of UNet code and can only receive primitive types or arrays of primitive types as parameters.

Following this logic, we created a class responsible for sending Commands (requests from the client to the server). Any action performed by the user that needed to be broadcasted to the network would be forwarded to this class and sent to the server. Similarly, we created a class responsible for sending RPCs from the server to the client. Although the RPC can be sent from anywhere from the server, centring it in a single class would simplify the process of migrating the networking system later.

The first prototypes of the game were developed using this system. This allowed us to implement all the mechanics of the game through several quick iterations. Once all the mechanics were coded, we decided that we could implement the skeleton for the authoritative server still in this project, which was not initially planned.

During this phase, all the game logic verification was made by the client itself, meaning there was no validation by an authoritative server, but the skeleton for this implementation was in

place (Figure 27). When it was time to transition to the authoritative server, the changes to the client code would be minimal.

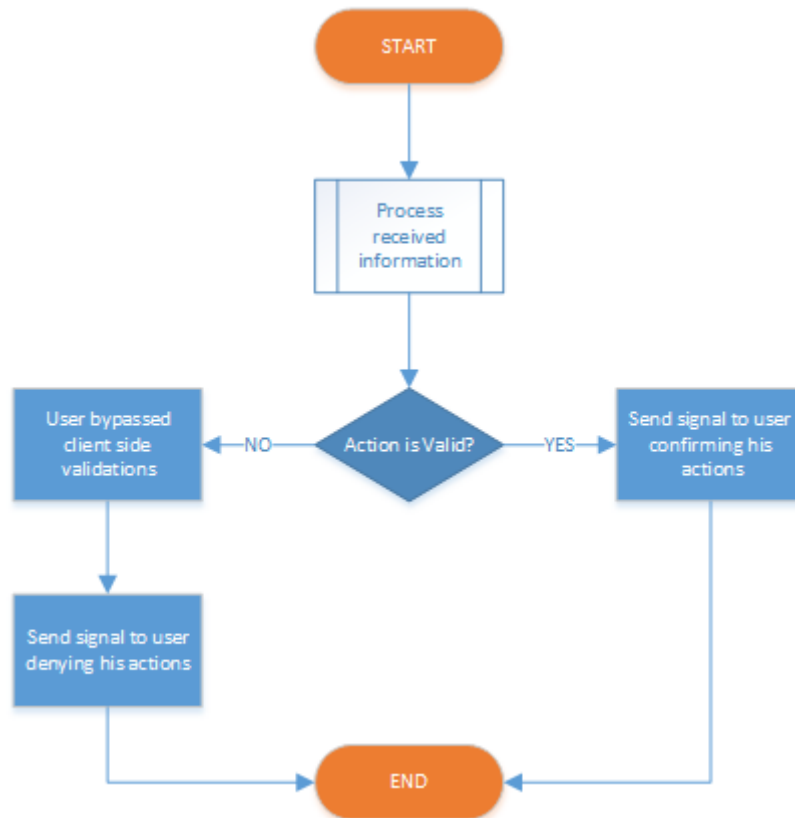


Figure 27 – Flowchart of server side validation. Client send the Command with relevant parameters to the server. Server verifies if those parameters are valid for the action the client wants to perform and answer accordingly by sending an RPC.

Since the server code and the client code were the same and all validations were made locally, it was very easy to access variables directly in the memory and hack the game. This was the main reason we wanted to implement an authoritative server. With all the logic being in the server, the user would not be able to hack the game this way. The process is very similar to what is done in websites: JavaScript performs client side validation, to reduce the unnecessary load on the server. After the validation is done, information is sent to the server and it revalidates everything, just in case the user altered the html or JavaScript on their side.

We searched for several solutions and the most appealing we found was Node.js. As mentioned before, the transition between UNet and Node.js was very simple, mainly because of the centralization of the networking communication. A decision that was made at that time was that the client would not have access to the database. All the SQL should be done by the server and then relayed to the client. This was another step towards increasing the security of the game. As with the UNet implementation, we did not code the server side validations, but left everything prepared for them.

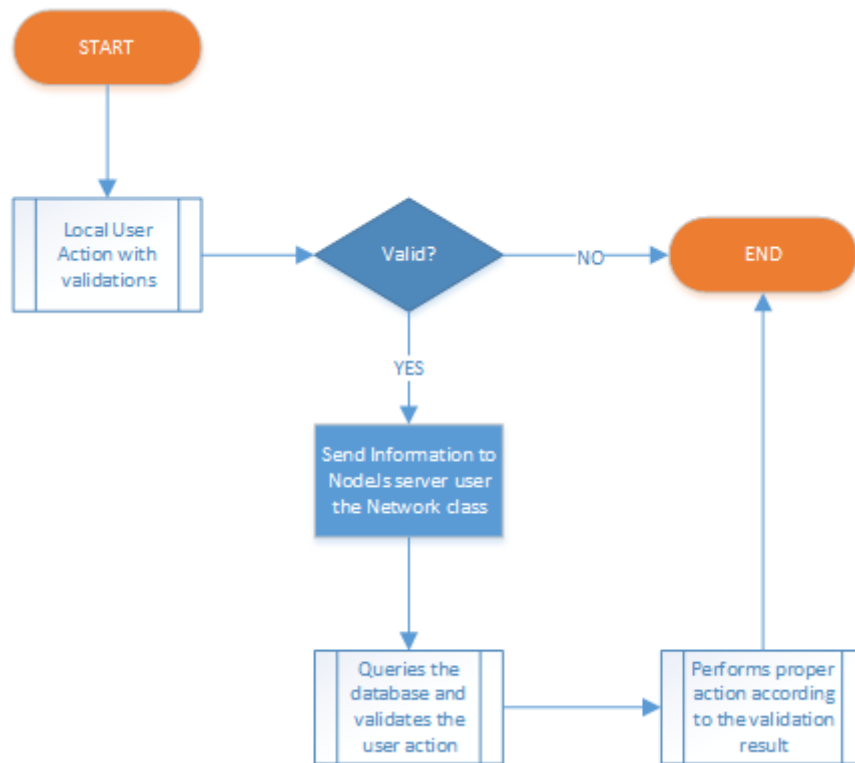


Figure 28 – Flowchart of the network pipeline using Node.js. It is very similar to UNet pipeline.

For deployment, we chose to host the server on Amazon’s AWS services, which provided us a virtual machine for free during one year [78]. This allowed us to play using two machines, even during the development phase. Since the server script can be interpreted both in Windows and Linux, this allowed us to use the same code to test in our local machine, running Windows, and then deploy online on the virtual machine, running on Ubuntu Server [79]. Once the server was deployed, we could just change a single line in U3D Editor to test the game with a local server. Then, perform the required changes and once the server was stable again, change everything back and redeploy online.

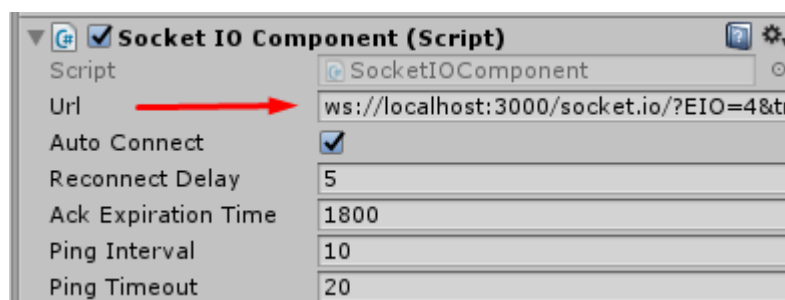


Figure 29 – Socket.IO component with the URL pointing to the server location. We could use localhost for testing and then deploy remotely to our Amazon EC2 machine.

Another feature we implemented in both networking systems was the matchmaking. We wanted to lay the foundation for a system that would allow us, in the future, to find games for players, matching them with opponents of similar skill. As a proof of concept, we just needed to match two players together. The algorithm used was very simple: if there are rooms with available open slots, join the first available, if there are not, create one and wait for more players. For the Node.js implementation, we have created a dedicated script to perform this logic, which will allow us to improve the matching algorithm in the future.

4.2.3 Game mechanics

4.2.3.1 Map Editor

The map is one of the most critical components of our game. Since it is a turn base, tiled strategy game, without a map there is no game. A considerable amount of time was spent planning and developing this system. We had started by trying to create an algorithm that would replicate the existing board game map in the game and that we could use to create random maps if we decided to. This led to several conceptual problems, as we would have to make sure that all the maps created automatically were evenly balanced for both teams. We then decided that the easiest way to reproduce the map layout of the board game was to create a map editor. This would allow us to visually place the content in the correct position and change its properties. It would also allow us to easily add more maps to the game in the future.

Even using a Map editor, there were several things to decide and plan before starting the implementation. We had to decide how a tile grid would be modelled, stored, loaded, archived and which properties each tile would require. To assist us on this task, we consulted Red Blob Games' blog on tile grids [40].

Following the guidelines on that blog, we ended up choosing two coordinate system for our map. An offset coordinate system for storing purposes (Figure 30-A) and an Axial coordinate system for real-time calculations (Figure 30-B). The offset coordinate system has the advantage that, for a rectangular map, columns (q) and rows (r), are the same as x and y in a Cartesian coordinate system. This means that for both storing and calculating the tile position in the game world, this coordinate system is the easiest to use. The axial system, on the other hand, had the advantage that its algorithms, such as rotations, distances, neighbours and lines are much simpler than the offset system. To use both systems, when loading the map from storage, we do a simple conversion, shown in Figure 30-C.

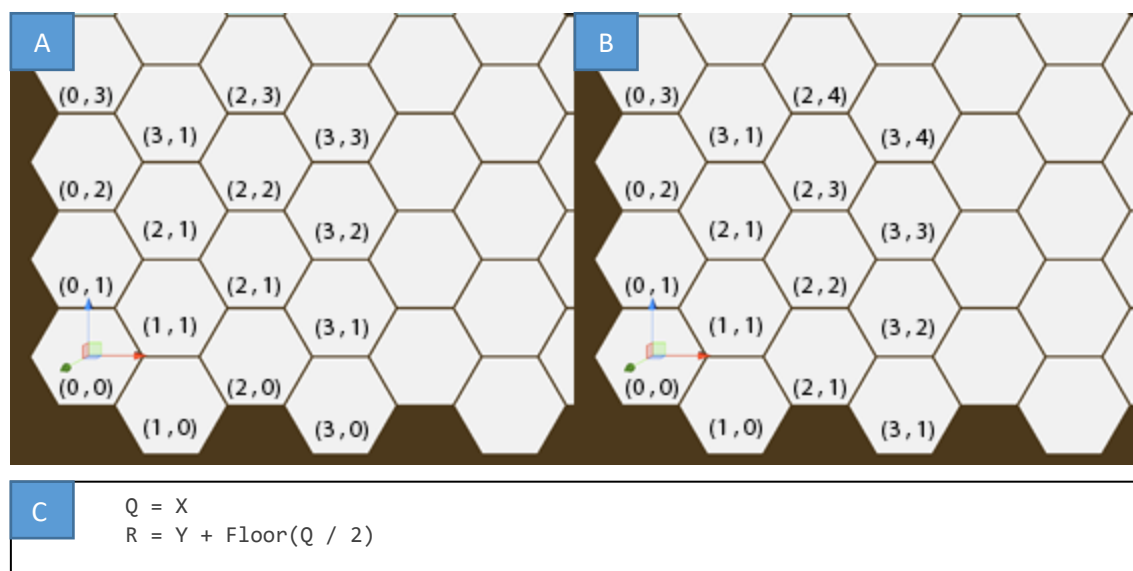


Figure 30 – A: offset coordinates. B: axial coordinates. C: Conversion from offset coordinate system (represented by X and Y) to axial coordinate system (represented by Q and R).

For storing the map, we used U3D's serialization feature, the scriptable object. The process of serializing and de-serializing these objects is automated by U3D, but it does not support the storage of matrices. To solve this, we used the Adapter design pattern and created a class, the HexList, that has only one property: a list of tiles. Each HexList represents a column, and each

node inside a HexList represents a single tile in the map. It is a complex solution, but it was the most efficient way to perform this process. The alternative would be to code our own serialization for this.

Once we could store and render all the tiles in their correct position, it was then time to start defining which properties each tile would have. The first thing was to give the user the ability to add or remove tiles. We decided that the best approach to do this, was to allow the users to only add or remove tiles from the four extremities of the map. For complex shapes, the user could set the information that the tile is not enabled and as such, it would not be rendered once the map was loaded in the game. As mentioned before, we extracted the requirements for each tile from the board game. Some of those requirements are irrelevant for the map building process, so we did not allow the user to change their value from the editor, such as if a tile is occupied or not. A very simple GUI was developed by us to allow the user to tweak the tile's properties as the ability to select several tiles for modification at the same time. Another quick tool added was the ability to use a toggle, so we could enable or disable tiles with only one click (Figure 31). Although the development of this editor took a considerable amount of time and it was not in the initial requirements, we think that it was worth it, since it allows us to quickly draw the map and change its properties. It is also a powerful tool for the future, when we decide to implement new maps for the game.



Figure 31 – Map editor tools developed to allow us to quickly change information about the map and the tiles. It can be used to add rows and columns to the extremities of the map and to change tile properties. Data is then stored in a Scriptable Object and is read and rendered in the game.

From the beginning, we decided that we would implement an A* heuristic for path finding and since this heuristic requires each tile to know their neighbours, we also created an algorithm to find all the tile's neighbours once these are loaded in the game and cache them in a list to use later. Disabled tiles were ignored.

With each tile knowing its neighbours, we could start implementing A* pathfinding. Once again, we consulted Red Blob Games' blog [40], in particular their implementation of A* pathfinding.

4.2.3.2 Abilities & Elemental Powers

The Abilities and Elemental Power systems were the most challenging aspects to implement. First, when developing the abilities in GGP, we made them to feel as unique as possible. Thus, close to none of the mechanics were reutilized when designing them, making the modelling process more arduous. Secondly, we wanted the code to be the most abstract and maintainable possible and reutilize as many algorithms as possible.

4.2.3.2.1 Abilities

The solution we found was to utilize C#'s inheritance feature and use an abstract class as a root for the ability system and make all individual abilities derive from this class and override all required methods. This abstract class works like an interface, with the advantage that we did not have to repeat code in the child classes and we could use the abstract class to define part of the algorithm. For instance, every ability has a cost, range, main target, main target restriction, number of targets and health modification (heal or damage), so this part of the information can be placed in the abstract class. We also used several static helper classes for repeated logic used in more than one ability. One example of this is the *MapMath* static class, responsible for all calculations regarding map and tiles, such as distance, path finding or populating tile neighbours.

Once the abstract class was defined, we created one class for each ability in the game, which allowed us to code different behaviours for each of them. There are a total of 53 abilities in the game, eight for each one of the six heroes plus, five for a hero that can copy other heroes' abilities (Table 4). Each ability uses the data stored in the local cache to calculate the available main targets, available secondary targets and if it can be used. There are three types of abilities: regular, cycles and counters. These abilities can only be used if the hero's team has enough resources and if all the required targets are in range. Regular and cycle abilities can only be used if the hero is playing. Counter abilities can only be used when the other team is playing and they usually cancel the ability being cast at the time. Some abilities have multiple choices, with different behaviours for each one. Cycle abilities permanently reserve the cost in resources while it is turned on, i.e., at the end of the round, the team does not regenerate the resources used to cast that ability.

Each ability belongs to one of three categories: Utility, Damage or Ultra. We call these categories polarities. As mentioned before, Utility abilities are used to give heroes mobility, support or defence capabilities, Damage are used to deal damage to other heroes and the Ultras are the signature move of the heroes (Table 4). When a hero unlocks a new ability, he can choose one ability of a polarity that matches his next available ability slot. The slots can have one or two polarities, varying from hero to hero, but the last slot is always of the type Ultra.

Table 4 – All abilities, with a short description of their mechanics and their polarity. Consult appendix 9.2.1 for the full version.

Name	Mechanic	Polarity
Ghora1	Deal 1 damage plus 1 for every 2-health missing.	Damage
Ghora2	Deal 1 damage plus 1 for every 3 current health.	Damage
Ghora3	Moves 3 tiles. Deal 1 damage around the final tile.	Utility
Ghora4	Redirects an ability to Ghora.	Utility
Ghora5	Reflects on ability back to the caster.	Utility
Ghora6	Deals 2 damage to enemies and heals 2 health to allies around Ghora.	Damage
GhoraU1	Ghora becomes immortal when activated. Spend resources instead of life when taking damage.	Ultra
GhoraU2	Intercepts one ability that would hit an ally.	Ultra
Kulla1	Deals 2 damage. If the target dies, deals 1 damage around the target.	Damage
Kulla2	Protects Kulla from the next damage ability when activated.	Utility
Kulla3	Places 2 traps in the map. Each trap deals 2 damage.	Damage
Kulla4	Deals 2 damage and moves the target 1 tile closer.	Damage
Kulla5	Move two tiles and apply Kulla element to targets on the final tile.	Utility
Kulla6	Deals 1 damage. If there is a Hero next to the target, it loses its next turn.	Utility
KullaU1	Deal 3 damage. Uses more resources to increase the range.	Ultra
KullaU2	Two targets: deals 3 damage to the closest, 2 damage to the other	Ultra
Lipp1	Cancels an ability. If Lipp was the target, deals 1 damage to the caster	Utility
Lipp2	Swap position with the target. Pushes or pulls everyone two tiles away in a two-tile range, depending on the user choice	Utility
Lipp3	Deals 3 damage. If the target dies, deals 2 more to another target.	Damage
Lipp4	Deals 1 damage. If the target dies, deals 1 more to another target.	Damage
Lipp5	Deals 2 damage to the target and 2 more to another target.	Damage
Lipp6	Reflects damage at the cost of resources if activated.	Utility
LippU1	Swap “Already Used” in every Lipp ability.	Ultra
LippU2	Copies an ultra from the target.	Ultra
Loper1	Swaps health with the target. Neither can exceed its maximum health.	Utility
Loper2	Can die in the place of a nearby ally.	Utility
Loper3	Deals 2 damage. Heals self for 2.	Damage
Loper4	Deals 2 damage. If the target dies, Loper can use 1 of the target ability for free.	Damage
Loper5	Transfers up to 3 health between two targets of the same team.	Utility
Loper6	Splits damage with nearby allies when they are attacked if activated.	Utility
LoperU1	Resurrects an ally with half health. Can use own health to pay cost.	Ultra
LoperU2	Controls a dead hero for one hero. Target abilities cost are paid with Loper health.	Ultra
Malik1	Cannot be target of abilities and can only use Utility abilities if activated. Abilities cost one more resource to cast.	Damage
Malik2	Deals 2 damage. If Malik has not unlocked Malik1, deals 1 more.	Damage
Malik3	Can move 1 tile before attacking and deal 1 damage. Deals 2 damage if Malik does not move.	Damage
Malik4	Cancels a damage ability or swaps “Already Used” in a target ability.	Utility
Malik5	Removes a trap from the map or heals Malik for 1 health.	Utility
Malik6	Hides every hero around Malik. Hidden hero cannot cast or be targeted by abilities.	Utility
MalikU1	Deals 3 damage plus 1 for each hero in a 2-tile range.	Ultra
MalikU2	Activates in one turn. Can kill any hero in range when it finishes activating.	Ultra
Wang1	Deals 2 damage to heroes in a frontal cone of Wang.	Damage
Wang2	Deals 1 damage and moves the target 2 tiles.	Utility
Wang3	Deals 2 damage. If the target dies, heals Wang for 2 health.	Damage
Wang4	Teleports to the first target and deals 2 damage. Teleports do the second and deals 1 damage.	Damage
Wang5	Redirects one ability to another target. Cannot return the ability to the caster.	Utility
Wang6	Swaps the position of 2 heroes.	Utility
WangU1	Deals 2 damage if the target has more than half health, otherwise kills the target.	Ultra
WangU2	Creates a clone that behaves as a regular hero when activated.	Ultra

Every user action must be verified before sending any information request to the server to execute the ability. If the user tries to add a tile as a target for an ability that requires a hero, or if he tries to attack someone that is out of his range, the verification system must detect the error and inform the user. Several processes were automated when there was no need of user input. For instance, if an ability would hit every target around the hero (as Ghora6 does), then the system automatically targets all of them without the need of extra input from the user.

There are several classes responsible for processing the mechanics of each ability. In addition to the previously mentioned general purpose static classes, there are also static classes specific for each hero. These helper classes are called `Execute'HeroName'Abilities.cs`. Following the execute ability pipeline, these classes receive all the information required to execute one ability (Figure 32 and Figure 33). While executing the mechanics, these classes pass information to DAP, responsible for the visualization, where animations are synchronized with the game mechanics. Once the ability finishes execution, combination system calculates all combinations generated by that ability. After the hero finishes its turn, all combinations are processed and Element Powers are drawn and processed.



Figure 32 – Lipp executing Lipp2. A: Lipp swaps position with its target. B: Lipp starts animation to push secondary targets away from her. C: Targets land on the first empty available tile for them and return to their idle animation.

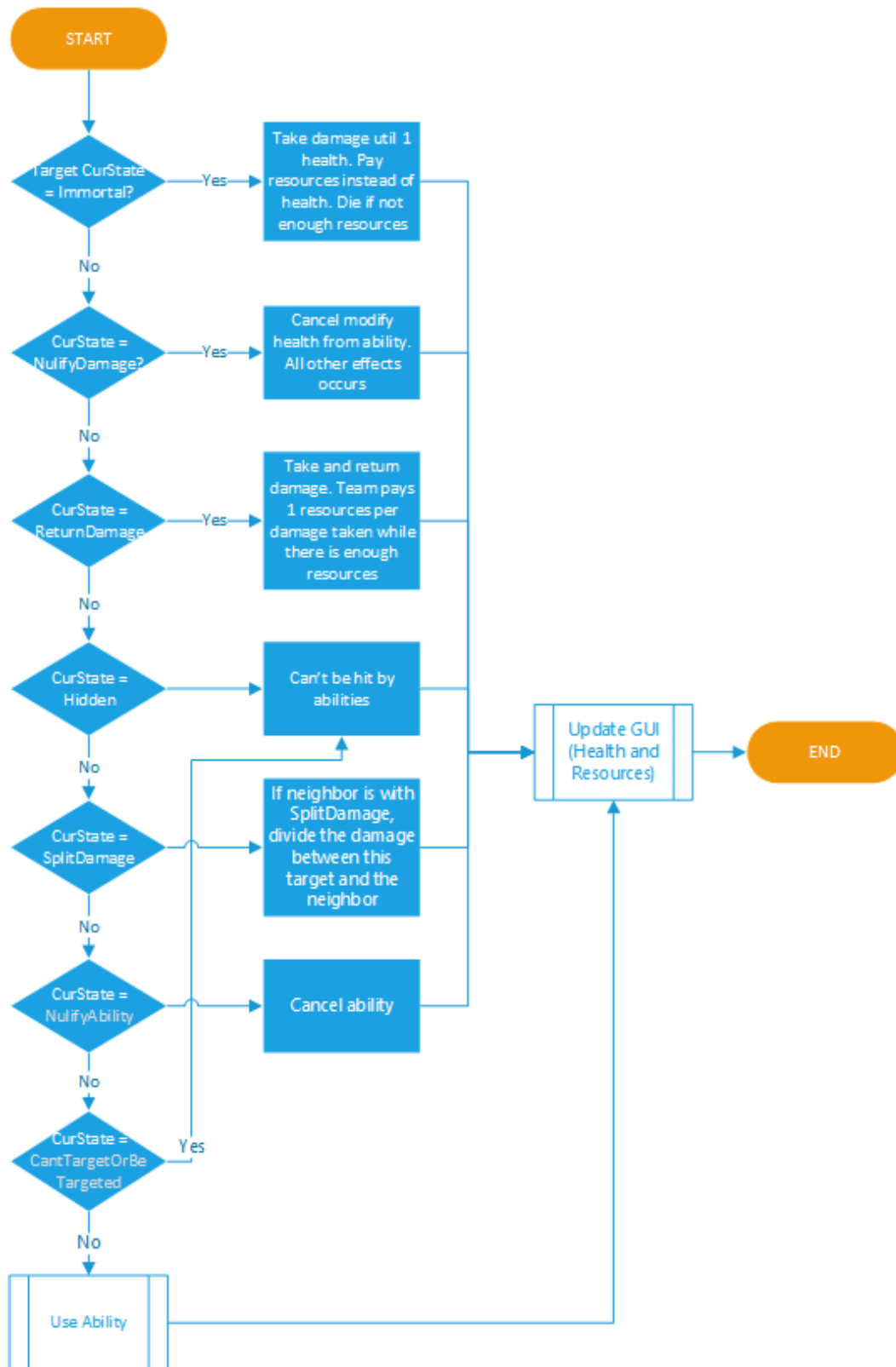


Figure 33 – Execute Ability pipeline. After validating all conditions for the usage of an ability, the game check if the ability will be executed normally or with exceptions.

4.2.3.2.2 Element Power

The EP follow the same organization as the abilities: there is an abstract class in the top of the hierarchy used to avoid code repetition and one concrete class for each EP. There are 55 EPs in the game, 11 for each element (Table 5). For each element, there is a sort of virtual deck, where all EPs of that element is stored and repeated per their rarity. Each deck has three of each common EP, two rare and one epic, once all EP in a deck has been used, the system resets all the cards of the deck, so they can be drawn again. The EP can, as the abilities, calculate its required targets and if they can be used. To process how to execute each element, there is also one class for each element called `Execute'ElementName'Power.cs`. As with the abilities, each EP might have more than one choice with different behaviours. When the combinations system informs the element power system that there are powers to be gained, this system draws a random power of the given element that has not yet been drawn.

Table 5 – All EP, with a short description of their mechanics and their play time. Consult appendix 9.2.2 for the full version.

Name	Mechanic	Play time
A1	Heals 1 health to the last playing hero of the player's team and 1 more to another.	Instant
A2	Heals 1 health to the last playing hero of the player's and 2 more to another.	Instant
A3	Heals 1 health to the last playing hero of the player's and 3 more to another.	Instant
A4	Heal 1 health for each damage ability the hero has. Minimum of 1.	Instant
A5	Heals 3 plus 1 for each hero of the same team nearby.	Instant
A6	Choose one: draw one Fire EP; draw one Wind EP; heals 1 health to a hero.	Instant
A7	Choose one: draw one Lightning EP; draw one Earth EP; heals 1 health to a hero.	Instant
A8	Heals 1 hero for health. Heals one more if there is a hero controlling water in the game.	Instant
A9	All heroes from the player team heal 1 health per damage and utility abilities the other team has and +2 for each ultra.	Instant
A10	Heals a hero for 1 plus the number of unlocked abilities the opposing team has unlocked.	Instant
A11	Heals 1 health to all heroes of the player team. Heals one more if the player has lost a base.	Instant
F1	Deals 1 damage to the last playing hero of the player's team and 1 to another.	Instant
F2	Deals 1 damage to the last playing hero of the player's team and 2 to another.	Instant
F3	Deals 1 damage to the last playing hero of the player's team and 3 to another.	Instant
F4	Deals 1 damage for each damage ability the hero has. Minimum of 1.	Instant
F5	Deal 3 damage plus, 2 for each hero of the same team in an adjacent tile.	Instant
F6	Choose one: draw a Water EP; draw a Lightning EP; deals 1 damage to a hero.	Instant
F7	Choose one: draw a Wind EP; draw an Earth EP; deals 1 damage to a hero.	Instant
F8	Deals 1 damage for each hero controlling Fire. Minimum of 1.	Instant
F9	Choose 1: restore 1 resource: Deal 1 damage to a hero.	Instant
F10	Deal 1 damage plus 1 for each additional ability the opposing team has unlocked.	Instant
F11	Deal 1 damage to all heroes of the opposing team. Deals 1 more if the player has lost a base.	Instant
R1	Use an ability for free.	Save for later
R2	Inverts the effect of the ability being played.	Save for later
R3	Steals 1 EP from the opposing team.	Save for later
R4	Reduces the cost of an ultra by 2.	Save for later
R5	Forces the opposing team to swap the last unlocked ability of the targeted hero.	Instant
R6	During one round, both teams can't use more abilities than the number of unlocked abilities of the one which has the least abilities.	Element Zone
R7	Choose 1: draw an Earth EP; draw a Fire EP; deal 1 damage to a Hero; Move a hero 3 tiles.	Instant
R8	Choose 1: draw a Water EP; draw a Wind EP; heal 1 health to a Hero; Move a hero 3 tiles.	Instant
R9	Choose 1: deal 1 damage to a character and a base; heal 1 health to a character or a base.	Instant

R10	Increase the cost of the opposing team's abilities by 1. Increases for one more if the opposing team has more abilities than the player.	Element Zone
R11	Unlocks 1 ability for every 3 abilities the opposing team has.	Instant
V1	Heroes inside the player base suffers the tower damage and are moved away of the area. If they re-enter, they suffer the damage again.	Element Zone
V2	Heroes are moved away from the base for 1 + (tower damage) tiles.	Save for later
V3	Move 2 heroes 2 tiles.	Instant
V4	After taking damage, hero moves away from the cast a number of tiles equal to the damage taken.	Element Zone
V5	Choose 1: draw a Water EP; draw an Earth EP; heals a tower by 1 health; Move 1 hero 3 tiles.	Instant
V6	Choose 1: draw a Lightning EP; draw a Fire EP; heals a tower by 1 health; Move 1 hero 3 tiles.	Instant
V7	Choose 1: Move 1 hero 3 tiles for each hero controlling wind in the game (Minimum of 3); Heal a tower for 1 health (Minimum of 1).	Instant
V8	During one round, heroes from the opposing team can't stay inside this team bases.	Element Zone
V9	The hero playing can move up to 6 tiles in their first movement phase.	Save for later
V10	Heroes from the player's team cannot be targeted by the other team.	Element Zone
V11	Move one hero up to 12 tiles.	Instant
T1	The player's heroes deal and take 1 more damage from towers.	Element Zone
T2	Deal 2 damage to a tower if there are two heroes from the player's team in a base.	Save for later
T3	Choose 1: after killing a tower, heal the hero to full health; the hero takes no damage the next time it attacks a tower.	Save for later
T4	The hero attacking the tower deals 2 more damage to it.	Save for later
T5	Choose 1: Draw a Lightning EP; draw a Wind EP; deal 1 damage to a tower.	Instant
T6	Choose 1: Draw a Water EP; draw a Fire EP; deal 1 damage to a tower.	Instant
T7	The hero attacking the tower deals 1 more damage for each hero controlling Earth in the game. Minimum of 1.	Save for later
T8	The hero attacking the tower deals 1 more damage and take 1 less damage.	Save for later
T9	During this round, heroes from the Player's team are immune to tower damage.	Element Zone
T10	If the player's heroes are inside the opposing base, they cannot be target of abilities.	Element Zone
T11	The hero currently playing can teleport to a base and deal 1 damage to it.	Save for later

Since this is a complex process to follow, we will provide a small example on how it works. Suppose that the user is current playing with Lipp hero and the game state is as shown in Figure 34. The hero selected is Lipp, from team Chronis. It is currently playing and it has moved once this turn. Since it only has moved once, it can still use abilities if the player wants to. The player then chooses to use the ability Lipp2. This ability allows the hero to choose a main target and swap positions with it. For the first choice of this ability, the target type is a hero and the player chooses to attack the other team's Kulla. The ability system then selects all secondary targets automatically for this ability and informs the player that he can use the ability. The player can now press "Use" to inform the system to use the ability and to propagate it through the network to the other player.



Figure 34 – Demonstration of using an ability (Lipp2). Selected hero is Lipp, from team Chorinis. It displays all information relevant to the currently selected ability and informs the player that he can use it.

After the swap is performed, it will then push or pull every hero in a two-tile radius by two tiles, depending if the player chose option 1 or 2. For the pushing algorithm, we extrapolate a straight line from the casting hero (Lipp in this case) and each of the targets and then, return the first empty tile that was two tiles away. If there were no tiles available, the target would not move. These extrapolation lines can be seen in Figure 35, painted in blue over the tiles.



Figure 35 – Extrapolation lines generated for each secondary target of Lipp 2 abilities. these lines pass through as many tiles as the 'distance' parameter received in function ExtrapolateHex.


```

internal static MonoHex ExtrapolateHex(MonoHex sourceHex, MonoHex finalHex,
int distance)
{
    var sourceVector = sourceHex.transform.position;
    var finalVector = finalHex.transform.position;
    var distInHexes = DistanceInHexes(sourceHex, finalHex);
    var t = 1f / (distInHexes);
    var desiredT = 1f + distance * t;
    var newVector3 = Vector3.LerpUnclamped(sourceVector, finalVector,
desiredT); // this draws a straight line, from point
source to final, with desiredT length
    var hex = MonoMap.First(mh => mh.transform.position == newVector3);

    return hex; // can be null if there are no valid hexes
}

```

Figure 36 – Code for extrapolating the first available tile in a given distance line.

Now if the user ends his turn, the combination system solves all combinations resulting from all abilities used this turn. In this example, only one ability was used that hit four targets (the main target was also a secondary target, so it was hit twice). It produced one neutral combination, two negatives and one positive combination (Figure 37).



Figure 37 – Execution of the EP gained after the execution of the ability Lipp2

5 Prototypes

This project had several key moments, that required us to guarantee that we could rely on a certain technology or that a game mechanic would work properly. There were three major milestones: connect and read the database from U3D, synchronize two game clients in a networked match and implement and propagate the game mechanics through the network.

5.1 Prototype 1

The first prototype we created was a simple U3D project that did a select query in a database and printed the result. From the knowledge acquired in previous projects, we knew that we could perform advanced SQL queries from C#, using MySQL Connector. So the only thing needed to verify was that we could connect U3D version of C# (Mono 2.0, which is based on .Net 2.0 [80]). To do this, we imported the mono version of the MySQL connector and placed the .dll files in U3D's plugins folder. By doing this, we could have access to all functions existing in the MySQL Connector library and by following the instructions on Oracle's Developer Guide, we were able to open a connection and perform a simple query [81]. Since the MySQL Connector, requires the usage of .NET functionalities not available on U3D's default subset, we had to change the API compatibility level from ".NET 2.0 subset" to ".NET 2.0" [82]. The knowledge gained during the development of this prototype was then used to implement all the queries for the balancing parameters and both in the Menus and In-game scenes.

5.2 Prototype 2

The second prototype was built to test how we could implement the game mechanics in the game. For this, we choose two abilities, an extremely simple (Ghora1) and the one we considered to be the most complex in the game (LoperU1). The goal of this prototyping was to verify if we could check all the required game rules to allow a player to use an ability and then execute that ability. The player had to be playing, his team had to have enough resources to use the ability, the ability could not have been used yet on that turn, the target had to be alive and in range. A rendering of this prototype can be found in Figure 38. Once all the criteria were met, the ability had to be executed, removing the amount of resources spent casting it and dealing with the ability's damage and secondary effects. We successfully built a prototype that could execute these abilities and, later used the architecture developed here to build the remaining abilities and element powers in game.

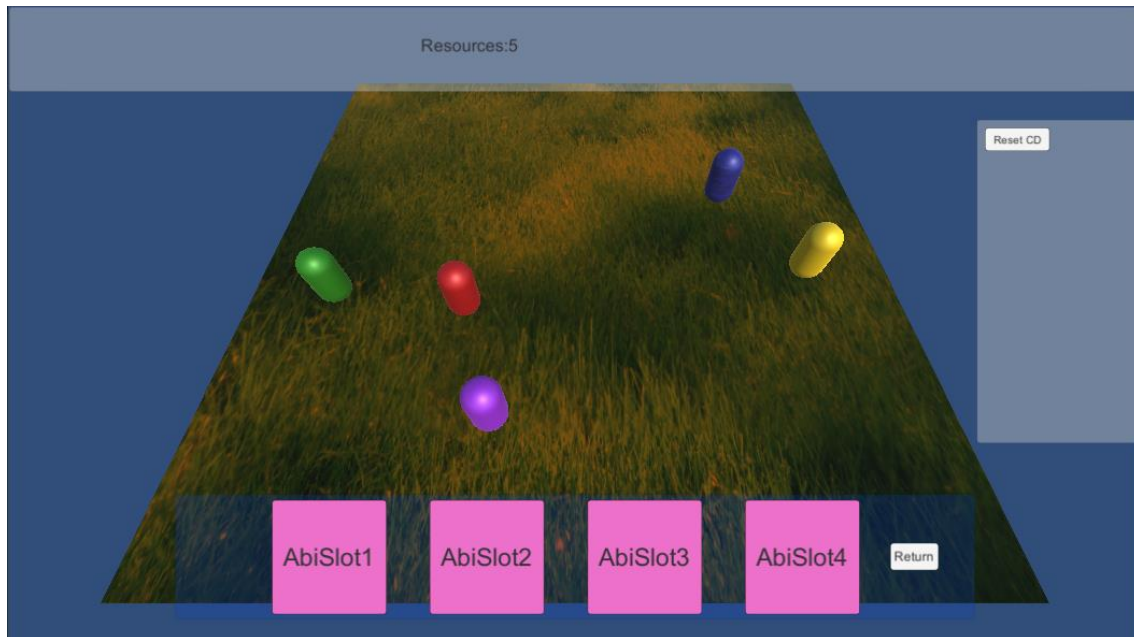


Figure 38 – Low fidelity prototype of the ability system. This was the second prototype of the game and it could be used to test abilities without the existence of a map.

5.3 Prototype 3

The last prototype was regarding the network connectivity. Several solutions were tried. First, we wanted to create a completely authoritative server and do the communication using Web Services. We implemented the first few steps towards this solution, but due to U3D's lack of compatibility with Web Services at the time, we had to abandon this route [74], [83]. Another solution attempted was the usage of PUN [37]. This looked promising at first, but the more we explored it, the more evident the problems with this solution became. Three major factors made us not use PUN: the official tutorials were outdated and incomplete, there was not API documentation at the time and the prices for more than 20 concurrent users were prohibitive to us [75]. We decided that we would use U3D's native networking for now and make the code easy to replace later, once U3D accepted web services or another solution was found for the authoritative server. Using the network pipeline described in section 4.1.3, we built a prototype of the movement logic. The player could start his turn and choose a tile to move to. Once he confirmed his action, his hero would be teleported to that tile. This prototype allowed us to understand how to architecture and improve the code and to propagate all the game mechanics to all clients. Once this last prototype was functional, the code from this project and DAP were combined. Using the pipeline developed in this project, we implemented all game abilities and element powers. A rendering of this prototype can be found in Figure 39.



Figure 39 – Prototype that allowed us to implement and test all game mechanics in the network.

5.4 Final prototype

After all intermediate prototypes, we then proceeded with the creation of the final version. This final prototype was the one used to do most of the playtests. All the game requirements are present in this final version of the game. There is only one major difference from the board game: this version of the game is 1 vs 1 instead of 3 vs 3. We found that having a 3 vs 3 game would make the experience much slower to the players, since, even if we set the turn time limit to 1 minute, the player would have to wait at least 5 minutes between plays.

This version of the prototype is prepared to support a free-to-play business model, where the player can use ingame currency to buy new content. This currency can be gained by playing the game or by buying it with real money. None of the systems for gaining the currency are present in the prototype, but the system to use the currency is.

In the menus section of the game, all functionalities were implemented (Figure 40). The player can create an account, which will be inserted into the database with the encrypted password, and then use the username and password to login into the game (Figure 40-A). Once inside the main menu, the player can go the Shop to buy a hero that is still locked for his account. The game checks the player's balance, checking the database, and if the player has enough Gold (in-game currency), then the purchase is confirmed and added to the database for persistency (Figure 40-B). The player can also create a team of three heroes. The team cannot have repeated heroes or elements. Pressing "Ready" confirms the selection (Figure 40-C). The player can also look for a game and select a game mode. For the final prototype, we only implemented the versus system, since it was sufficient for a proof of concept product, but in the future, it will be possible to choose from any of the other game modes. Once a game mode is selected, the database is updated with the hero team and the game starts to look for other players to play (Figure 40-D).



Figure 40 – Menu scenes with functionality implemented. A – Login screen. It is possible to login and create a new account from this screen. B – Shop screen. In the shop the player can buy heroes still locked in his account. Heroes are bought with in-game currency. C – Team Selection screen. From here, the player can choose all heroes in a team and change their element. D – Find a Game screen. In this prototype, only the Versus game mode was implemented.

Once in the game, the player can do the same actions as in the GGP and all of them are executed properly and propagated through the network to the other player. The player can move, use abilities on heroes, attack towers and use EPs (Figure 41).

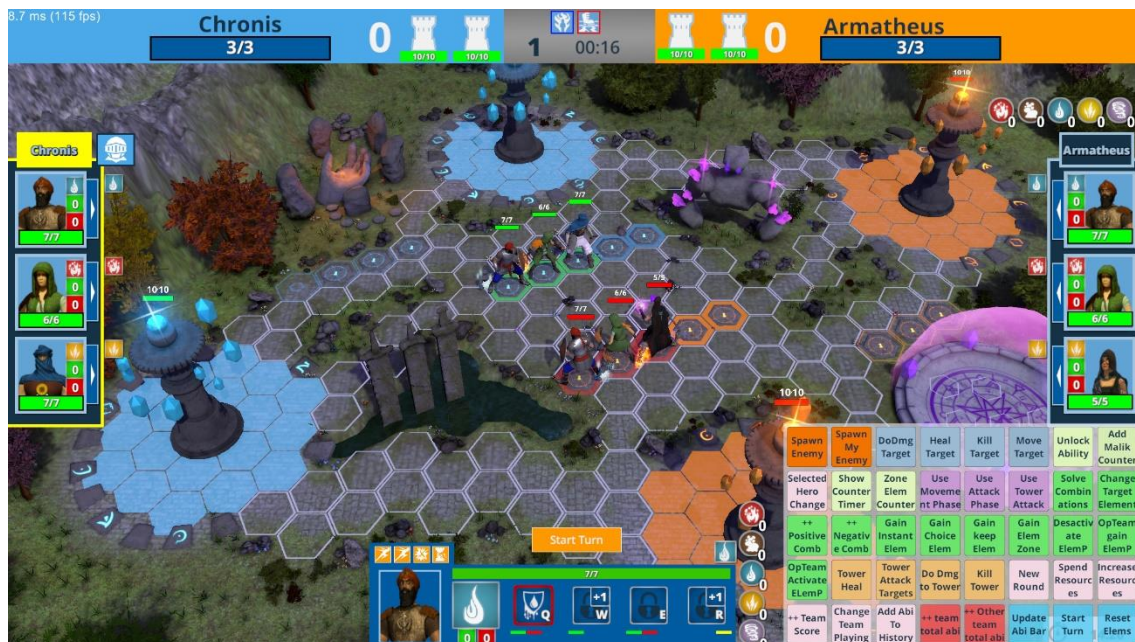


Figure 41 - Final prototype with all mechanics from the GGP implemented.

To assist with the local testing, we created a panel with functions that emulate the behaviour of the remote player. This panel can be seen in Figure 42 . This allowed us to quickly test the multiplayer functionalities inside U3D editor without the need to compile and deploy the game.

Spawn Enemy	Spawn My Enemy	DoDmg Target	Heal Target	Kill Target	Move Target	Unlock Ability	Selected Hero Change
Use Movement Phase	Use Attack Phase	Use Tower Attack	Solve Combinations	Change Target Element	++ Positive Comb	++ Negative Comb	Gain Instant Elem
Gain Choice Elem	Gain keep Elem	Gain Elem Zone	Deactivate ElemP	OpTeam gain ElemP	OpTeam Activate ELemP	Tower Heal	Tower Attack Targets
Do Dmg to Tower	Kill Tower	New Round	Spend Resources	Increase Resources	++ Team Score	Change Team Playing	Add Abi To History
++ team total abi	++ Other team total abi	Update Abi Bar	Start Turn	Reset Elems	Release Zombie		

Figure 42 – Actions that can be performed by the game state altering feature.

6 Playtests

At this time, it is important to remind the reader that this was not a game developing project and, as such, the playtests described in this section were not the usual playtests sessions encountered in the game development cycle. The goal of this project was to do a digital version of an already developed board game, so the goal of these tests was to identify bugs, evaluate the impact of identified problems, retest to evaluate if the problems were fixed and evaluate the overall user experience [84].

The first few hundred sessions were performed by us. While implementing the systems and mechanics, we performed small tests to evaluate if the code was performing as intended. Thanks to U3D's capability of quickly loading the game inside the editor, without the need of compiling the whole project to test it, we could perform dozens of small tests every day. This was an ongoing process during the development phase.

Once all the mechanics and systems were in place, we combined both teams from this project and started to perform long sessions of gameplay. The intend of these tests was to see how all the systems performed during an entire game session. Some final adjustments were done in this phase, as we identified small problems and inconsistencies.

The next phase was to test the game with people who knew the board game but did not work in the development of the board game. This was the most crucial phase of testing, as users who are not involved with the development process are usually the best ones to find unthinkable problems. To assist us in identifying the errors, we used both a think aloud method and recorded the screen of the game. Countless bugs were found and fixed during this phase. During a period of four weeks, we repeated the same cycle: playtest in the morning, fixing the errors in the afternoon. We did this until the number of bugs found per session was minimal. During this phase, we identified a problem with Kulla's trap ability (Kulla3) and we were not able to fix the problem. In the board game, Kulla would place two traps: one real trap, that would deal damage, and a bluff trap, that would just stop the hero's movement. Unfortunately, during the development of the board game we did not realise that this ability had some inconsistencies with the way it worked when compared with other abilities, so we had to change it to the current version (appendix 9.2.1). A complete list of the bugs identified during these playtests sessions can be found in appendix section 9.5.

For the final phase of testing, we invited gamers to test our game. The requirement for this phase was that the user had to be familiar with the game genre. The goal was to see if an unfamiliar user could use the software without encountering too many bugs. We did six matches, with a total of twelve users. During one match, an unidentified bug occurred at the beginning of the session, which made the game unplayable. All the other matches performed as expected, with minor bugs appearing, but all of them were able to use the game for more than one hour without any game breaking bugs.

7 Conclusion

This project has the objective of adapting a Board game to a digital game. The port was done using Unity3D. We had to design the architecture of the project, including the database, data structures and networking. It was also part of our responsibilities to adapt all the mechanics from the board game to the digital game.

The development of this project was a huge challenge and step up in difficulty from everything we had done during the Master's degree. It was an extremely ambitious project and to be able to continue the development and deployment of this product, ideally, at least two more developers should be added to the group.

The planning of the GIP was probably the biggest mistake we made. Before the DAP and MTP teams started to work separately, it would have been better if the architecture of the GIP was completely defined. By not doing this, the DAP team had to improvise and use a dummy set of data structures, which cost them time, and then, when we got to that part of the project, we had to redefine the data structures without breaking DAP's work. This was time consuming and unnecessary with better planning.

During the development of the MTP, research was done to find out how games are developed by the industry, how the software architecture of games is performed and how they solve the most common programming problems. The biggest finding on this topic is that this kind of information is extremely hard to find and usually it is written in such an abstract level that it is hard to comprehend and adapt to other projects. We found it more productive to adapt the regular software development process to develop the game.

We were responsible for architecting and programing all the systems required to adapt the GGP to a digital game. This included gathering the technologies that could help us on this process and develop all the systems, as networking, heroes, abilities, element powers, account and many others.

Despite all the difficulties, we were able to successfully implement everything we proposed in addition to adding the skeleton structure of the authoritative server. This experience was very productive, both on a personal and professional level. It was an eye-opening experience to see how the knowledge acquired during several years of courses at Universidade da Madeira, both in the Bachelor's and in the Master's degree, worked to allow us to complete this project. The fact that we developed a full game, from its concept, to its digital implementation, gave us a priceless experience that will be very helpful when applying for a position in the video game industry.

We were able to implement the systems for the user account, both in the database and in the game client. The player can create an account, login, save several preferences and buy heroes from the shop. All modifications to the user account done in the game were saved in the database for persistency.

Almost all mechanics existing in the board game were successfully translated to the digital game. The few that we were not able to adapt were because the mechanic itself was broken and we did not realise that during the board game testing phase (as it happened with Kulla3). All the remaining 47 abilities were implemented, as well as the 6 heroes and the 55 EPs. Both the abilities and the EPs are easy to expand, thanks to the abstractions used in the GIP. Also, for

balancing, they can be adjusted only by changing values in the database, without the need to recompile and redistribute the game.

The networking system was also implemented with success. We managed to connect several players to our server and allow them to play against each other, in rooms of two players. Each player only exchanges information with other players in the same room, so parallel matches are possible without any interference.

For future work, we would like to continue the development of the game by finishing the implementation of the authoritative server and by adding more content (heroes and maps) to the game.

8 References

- [1] T. Vieira, "Keepers of Intheris - Aesthetic and History," Universidade da Madeira, 2017.
- [2] R. Lauer, "The Elemental Tetrad of Games | Cleveland Institute of Art College of Art | 800.223.4700," 2015. [Online]. Available: <http://www.cia.edu/blog/2015/04/the-elemental-tetrad-of-games>. [Accessed: 07-Nov-2016].
- [3] ESA, "Game : Improving The economy," 2015.
- [4] A. N. Overview and O. F. Trends, "2016 Global Games Market Report," no. June, pp. 1–24, 2016.
- [5] J. N. Board, P. Awaits, N. Using, G. Scintigraphy, and C. R. Models, "What's Inside," *In Vitro*, vol. 27, no. 1, 2010.
- [6] G. Singer, "The History of the Modern Graphics Processor - TechSpot," 2013. [Online]. Available: <http://www.techspot.com/article/650-history-of-the-gpu/>. [Accessed: 07-Nov-2016].
- [7] J. Garger, "The History and Evolution of Computer Sound Cards," 2011. [Online]. Available: <http://www.brighthub.com/computing/hardware/articles/60555.aspx>. [Accessed: 07-Nov-2016].
- [8] J. Norman, "NIMATRON: An Early Electromechanical Machine to Play the Game of Nim (1940)." [Online]. Available: <http://www.historyofinformation.com/expanded.php?id=4472>.
- [9] Smithsonian, "The Brown Box, 1967–68 | National Museum of American History." [Online]. Available: http://americanhistory.si.edu/collections/search/object/nmah_1301997.
- [10] G. A. Sarcone, "Nim game." [Online]. Available: http://www.archimedes-lab.org/game_nim/nim.html.
- [11] Smithsonian, "Magnavox Odyssey Video Game Unit, 1972 | National Museum of American History." [Online]. Available: http://americanhistory.si.edu/collections/search/object/nmah_1302004.
- [12] AtariAge, "AtariAge - Atari 2600 History." [Online]. Available: <http://www.atariage.com/2600/>. [Accessed: 07-Nov-2016].
- [13] N. Oxford, "Ten Facts about the Great Video Game Crash of '83 - IGN," 2011. [Online]. Available: <http://www.ign.com/articles/2011/09/21/ten-facts-about-the-great-video-game-crash-of-83>. [Accessed: 07-Nov-2016].
- [14] R. Lambie, "The 1983 videogame crash: what went wrong, and could it happen again? | Den of Geek," 2013. [Online]. Available: <http://www.denofgeek.com/games/24531/the-1983-videogame-crash-what-went-wrong-and-could-it-happen-again>. [Accessed: 14-Nov-2016].
- [15] A. Cunningham, "The NES turns 30: How it began, worked, and saved an industry | Ars Technica," 2013. [Online]. Available: <http://arstechnica.com/gaming/2013/07/time-to-feel-old-inside-the-nes-on-its-30th-birthday/>. [Accessed: 14-Nov-2016].
- [16] B. Gladstone, "How Nintendo Saved the Video Game Industry - On The Media - WNYC." [Online]. Available: <http://www.wnyc.org/story/133033-how-nintendo-saved-the>

video-game-industry/. [Accessed: 17-Nov-2016].

- [17] Statista, "Video game, gaming industry revenue | Statista," 2016. [Online]. Available: <https://www.statista.com/statistics/278181/video-games-revenue-worldwide-from-2012-to-2015-by-source/>. [Accessed: 14-Nov-2016].
- [18] T. DiChristopher, "Digital gaming sales hit record \$61 billion in 2015: Report," 2016. [Online]. Available: <http://www.cnbc.com/2016/01/26/digital-gaming-sales-hit-record-61-billion-in-2015-report.html>. [Accessed: 14-Nov-2016].
- [19] J. van Dreunen, "PC trumps mobile, console in booming \$61bn digital games market," *Games industry*, 2016. [Online]. Available: <http://www.gamesindustry.biz/articles/2016-01-26-pc-trumps-mobile-console-in-booming-usd61bn-digital-games-market>. [Accessed: 14-Nov-2016].
- [20] T. Chatfield, "Videogames now outperform Hollywood movies | Technology | The Guardian," 2009. [Online]. Available: <https://www.theguardian.com/technology/gamesblog/2009/sep/27/videogames-hollywood>. [Accessed: 14-Nov-2016].
- [21] MPAA, "2012 Theatrical Statistics Summary," *Mpa*, p. 28, 2015.
- [22] C. Pereira, "2015's Top Digital Games Lists Include League of Legends, Call of Duty - GameSpot." [Online]. Available: <http://www.gamespot.com/articles/2015s-top-digital-games-lists-include-league-of-le/1100-6434142/>. [Accessed: 17-Nov-2016].
- [23] A. Rollings and E. Adams, *Andrew Rollings and Ernest Adams on Game Design*. New Riders, 2003.
- [24] A. Rollings and E. Adams, *Fundamentals of Game Design*. Prentice Hall, 2006.
- [25] Hasbro, "Risk Game | Toys for Kids | Risk." [Online]. Available: <http://www.hasbro.com/en-us/product/risk-game:2C7C6F52-5056-9047-F5DD-EB8AC273BA4C>. [Accessed: 20-Jan-2017].
- [26] Wikipedia, "Strategic Studies Group." [Online]. Available: https://en.wikipedia.org/wiki/Strategic_Studies_Group. [Accessed: 02-Apr-2017].
- [27] Wikipedia, "Westwood Studios." [Online]. Available: https://en.wikipedia.org/wiki/Westwood_Studios. [Accessed: 02-Apr-2017].
- [28] F. Herbert, "The Official Dune Website." [Online]. Available: <http://www.dunenovels.com/>. [Accessed: 20-Jan-2017].
- [29] P. Tassi, "Riot Games Reveals 'League of Legends' Has 100 Million Monthly Players." [Online]. Available: <http://www.forbes.com/sites/insertcoin/2016/09/13/riot-games-reveals-league-of-legends-has-100-million-monthly-players/#5e43571210b1>. [Accessed: 17-Nov-2016].
- [30] Richard, "Video Game Development Process - How2Become." [Online]. Available: <https://www.how2become.com/blog/video-game-development-process/>. [Accessed: 28-Dec-2016].
- [31] IGN, "The Game Production Pipeline: Concept to Completion - IGN." [Online]. Available: <http://www.ign.com/articles/2006/03/16/the-game-production-pipeline-concept-to-completion?page=1>. [Accessed: 28-Dec-2016].
- [32] I. Sommerville, *Software Engineering*. 2010.

- [33] J. Provost, "The Game Designer: Kickstarter and Feature Creep." [Online]. Available: <http://thegamedesigner.blogspot.pt/2013/07/kickstarter-and-feature-creep.html>. [Accessed: 25-Jan-2017].
- [34] StackOverflow, "How is game development different from other software development? - Game Development Stack Exchange." [Online]. Available: <http://gamedev.stackexchange.com/questions/9074/how-is-game-development-different-from-other-software-development>. [Accessed: 28-Dec-2016].
- [35] Wikipedia, "Waterfall Model." [Online]. Available: https://en.wikipedia.org/wiki/Waterfall_model. [Accessed: 20-Jan-2017].
- [36] Wikipedia, "Agile Software Development." [Online]. Available: https://en.wikipedia.org/wiki/Agile_software_development. [Accessed: 21-Jan-2017].
- [37] ExitGames, "Photon Unity 3D Networking Framework SDKs and Game Backend | Photon: Multiplayer Made Simple." [Online]. Available: <https://www.photonengine.com/en/PUN>. [Accessed: 08-Dec-2016].
- [38] Unity, "Unity - Manual: Multiplayer and Networking." [Online]. Available: <https://docs.unity3d.com/Manual/UNet.html>. [Accessed: 08-Dec-2016].
- [39] Unity, "Unity - Introduction to a Simple Multiplayer Example." [Online]. Available: <https://unity3d.com/learn/tutorials/topics/multiplayer-networking/introduction-simple-multiplayer-example?playlist=29690>. [Accessed: 08-Dec-2016].
- [40] Red Blob Games, "Hexagonal Grids." [Online]. Available: <http://www.redblobgames.com/grids/hexagons/>. [Accessed: 08-Dec-2016].
- [41] Microsoft, "Agile, Git, CI with TFS | Team Foundation Server." [Online]. Available: <https://www.visualstudio.com/tfs/>. [Accessed: 14-Dec-2016].
- [42] Microsoft, "Connect to team projects | Team Service & TFS." [Online]. Available: <https://www.visualstudio.com/en-us/docs/connect/connect-team-projects>. [Accessed: 14-Dec-2016].
- [43] J. Ward, "What is a Game Engine? - GameCareerGuide.com," 2008. [Online]. Available: http://www.gamecareerguide.com/features/529/what_is_a_game_.php. [Accessed: 21-Nov-2016].
- [44] M. Enger, "Game Engines: How do they work?," 2013. [Online]. Available: <http://www.giantbomb.com/profile/michaelenger/blog/game-engines-how-do-they-work/101529/>. [Accessed: 21-Nov-2016].
- [45] S. Zerbst and O. Düvel, *3D Game Engine Programming*. Premier Press, 2004.
- [46] Epic Games, "What is Unreal Engine 4." [Online]. Available: <https://www.unrealengine.com/what-is-unreal-engine-4>. [Accessed: 21-Nov-2016].
- [47] Crytek, "CRYENGINE | The complete solution for next generation game development by Crytek." [Online]. Available: <https://www.cryengine.com/>. [Accessed: 21-Nov-2016].
- [48] Unity, "Unity - Game Engine." [Online]. Available: <https://unity3d.com/>. [Accessed: 21-Nov-2016].
- [49] Unity, "Asset Store." [Online]. Available: <https://www.assetstore.unity3d.com/en/>. [Accessed: 23-Nov-2016].
- [50] R. Fortin, "Gamasutra: Raphael Fortin's Blog - 5 reasons why Unity is better for learning

- game development than Unreal Engine,” 2015. [Online]. Available: http://www.gamasutra.com/blogs/RaphaelFortin/20150302/237665/5_reasons_why_Unity_is_better_for_learning_game_development_than_Unreal_Engine.php. [Accessed: 23-Nov-2016].
- [51] Microsoft, “What Is Managed Code?” [Online]. Available: [https://msdn.microsoft.com/en-us/library/windows/desktop/bb318664\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/bb318664(v=vs.85).aspx). [Accessed: 23-Nov-2016].
 - [52] Unity, “Unity - Store.” [Online]. Available: <https://store.unity.com/>. [Accessed: 29-Dec-2016].
 - [53] EpicGames, “Unreal Engine Frequently Asked Questions.” [Online]. Available: <https://www.unrealengine.com/faq>. [Accessed: 21-Nov-2016].
 - [54] Crytek, “CRYENGINE | Crytek.” [Online]. Available: <https://www.cryengine.com/get-cryengine/memberships/>. [Accessed: 21-Nov-2016].
 - [55] StackExchange, “architecture - Game engine and data driven design - Game Development Stack Exchange.” [Online]. Available: <http://gamedev.stackexchange.com/questions/17331/game-engine-and-data-driven-design>. [Accessed: 05-Dec-2016].
 - [56] A. Monnappa, “The Rise of NoSQL and Why it Should Matter to You | Simplilearn.” [Online]. Available: <https://www.simplilearn.com/rise-of-nosql-and-why-it-should-matter-to-you-article>. [Accessed: 05-Dec-2016].
 - [57] R. Bourdon, “WampServer, la plate-forme de développement Web sous Windows - Apache, MySQL, PHP.” [Online]. Available: <http://www.wampserver.com/>. [Accessed: 05-Dec-2016].
 - [58] Oracle, “MySQL :: MySQL Workbench.” [Online]. Available: <http://www.mysql.com/products/workbench/>. [Accessed: 05-Dec-2016].
 - [59] Oracle, “MySQL :: MySQL Connectors.” [Online]. Available: <https://www.mysql.com/products/connector/>. [Accessed: 05-Dec-2016].
 - [60] F. Bevilacqua, “Building a Peer-to-Peer Multiplayer Networked Game.” [Online]. Available: <https://gamedev.tutsplus.com/tutorials/building-a-peer-to-peer-multiplayer-networked-game--gamedev-10074>. [Accessed: 29-Dec-2016].
 - [61] Gaffer, “Gaffer on Games | What every programmer needs to know about game networking.” [Online]. Available: <http://gafferongames.com/networking-for-game-programmers/what-every-programmer-needs-to-know-about-game-networking/>. [Accessed: 28-Dec-2016].
 - [62] NodeJs, “Node.js.” [Online]. Available: <https://nodejs.org/en/>. [Accessed: 29-Dec-2016].
 - [63] Unity, “Unity - Manual: The High Level API.” [Online]. Available: <https://docs.unity3d.com/Manual/UNetUsingHLAPI.html>. [Accessed: 03-Jan-2017].
 - [64] B. Pluszczewska, “9 Famous Apps Built with Node.js | Blog Brainhub.eu.” [Online]. Available: <https://brainhub.eu/blog/2016/05/30/9-famous-apps-using-node-js/>. [Accessed: 06-Jan-2017].
 - [65] Quora, “(2) What are the biggest websites built with Node.js on the server side? - Quora.” [Online]. Available: <https://www.quora.com/What-are-the-biggest-websites-built-with-Node-js-on-the-server-side>. [Accessed: 06-Jan-2017].

- [66] Tutorioalspoint, "Node.js Introduction." [Online]. Available: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm. [Accessed: 07-Feb-2017].
- [67] Quora, "(2) Why is Node.js becoming so popular? - Quora." [Online]. Available: <https://www.quora.com/Why-is-Node-js-becoming-so-popular>. [Accessed: 06-Jan-2017].
- [68] A. Zanzir, "Unity Multiplayer Game Development with Node | Pluralsight." [Online]. Available: <https://www.pluralsight.com/courses/unity-multiplayer-game-dev-node-2454>. [Accessed: 06-Jan-2017].
- [69] F. Panettieri, "Socket.IO for Unity - Asset Store." [Online]. Available: <https://www.assetstore.unity3d.com/en/#!/content/21721>. [Accessed: 06-Jan-2017].
- [70] Nuclearly Potent Moonshine, "npm." [Online]. Available: <https://www.npmjs.com/>. [Accessed: 06-Jan-2017].
- [71] Ecma International, "Welcome to Ecma International." [Online]. Available: <http://www.ecma-international.org/default.htm>. [Accessed: 06-Jan-2017].
- [72] N. C. Zakas, "Read Understanding ECMAScript 6 | Leanpub." [Online]. Available: <https://leanpub.com/understandings6/read>. [Accessed: 06-Jan-2017].
- [73] Kangax, "ECMAScript 6 compability table." [Online]. Available: <https://kangax.github.io/compat-table/es6/#node>. [Accessed: 06-Jan-2017].
- [74] Nosfsos, "Unity3D freezes when running a project with webservices for the second time - Unity Answers." [Online]. Available: <http://answers.unity3d.com/questions/1149928/unity3d-freezes-when-running-a-project-with-webser.html>. [Accessed: 15-Jan-2017].
- [75] ExitGames, "Photon Realtime Pricing Plans | Photon: Multiplayer Made Simple." [Online]. Available: <https://www.photonengine.com/en/Realtime/Pricing#plan-20>. [Accessed: 05-Jan-2017].
- [76] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
- [77] Blue Raja, "A Heap-Based C# Priority Queue Optimized for A* Pathfinding — One Man's Trash is Another Man's Blog."
- [78] Amazon, "Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS." [Online]. Available: https://aws.amazon.com/ec2/?nc2=h_m1. [Accessed: 06-Jan-2017].
- [79] Canonical Ltd., "Ubuntu Server." [Online]. Available: <https://www.ubuntu.com/download/server>. [Accessed: 06-Jan-2017].
- [80] Domino-Studios, "What is the version of .net in Unity 5? - Unity Answers." [Online]. Available: <http://answers.unity3d.com/questions/924021/what-is-the-version-of-net-in-unity-5.html>. [Accessed: 15-Jan-2017].
- [81] Oracle, "MySQL :: MySQL Connector/Net Developer Guide :: 5 Connector/Net Programming." [Online]. Available: <https://dev.mysql.com/doc/connector-net/en/connector-net-programming.html>. [Accessed: 15-Jan-2017].
- [82] Unity, "Unity - Manual: Standalone Player Settings." [Online]. Available: <https://docs.unity3d.com/Manual/class-PlayerSettingsStandalone.html>. [Accessed: 15-

Jan-2017].

- [83] giano574, "WebService WebMethod in Unity "timing out"? | Unity Community." [Online]. Available: <https://forum.unity3d.com/threads/webservice-webmethod-in-unity-timing-out.322031/>. [Accessed: 15-Jan-2017].
- [84] Centercode, "Beta Testing Objectives | Centercode." [Online]. Available: <https://www.centercode.com/beta/goals/>. [Accessed: 15-Jan-2017].
- [85] Riot Games, "League of Legends." Riot Games, 2009.

9 Appendices

9.1 GANTT

Table 6 – GANTT chart

Task Name	Duration	Start	Finish	Predecessors	Resource Names
Navegação de Menus com cliente ligado ao servidor	165 days	Mon 05/10/15	Fri 20/05/16		
Estrutura de Dados	130 days	Mon 05/10/15	Fri 01/04/16		
Criação do UML conceptual de classes	130 days	Mon 05/10/15	Fri 01/04/16		
ED GUI	100 days	Mon 05/10/15	Fri 19/02/16		Tatiana & Yuri
ED InGame	15 days	Mon 29/02/16	Fri 18/03/16		Tatiana;Yuri
Habilidades	30 days	Mon 15/02/16	Fri 25/03/16		Yuri
Elements ED	5 days	Mon 28/03/16	Fri 01/04/16		Yuri
Base de dados	110 days	Mon 19/10/15	Fri 18/03/16		
Criação diagram ER	15 days	Mon 19/10/15	Fri 06/11/15		Tatiana;Yuri
Ligação da base de dados ao Unity	5 days	Mon 23/11/15	Fri 27/11/15		Yuri
Estruturação das conexões as bases de dados (abstrações)	45 days	Mon 30/11/15	Fri 29/01/16	10;9	Tatiana;Yuri
Implementação base de dados (Selects)	30 days	Mon 04/01/16	Fri 12/02/16		
Copia local da base de dados	30 days	Mon 04/01/16	Fri 12/02/16		Yuri
Player	30 days	Mon 04/01/16	Fri 12/02/16		Tatiana;Yuri
Implementação base de dados (Inserts e Updates)	10 days	Mon 07/03/16	Fri 18/03/16		Yuri
GUI	165 days	Mon 05/10/15	Fri 20/05/16		
Navegação	130 days	Mon 05/10/15	Fri 01/04/16		
Árvore de menus	25 days	Mon 05/10/15	Fri 06/11/15		Tatiana
Conceito GUI	40 days	Mon 09/11/15	Fri 01/01/16	18	Tatiana
Log de Tarefas	15 days	Mon 02/11/15	Fri 20/11/15		Tatiana
Guião	5 days	Mon 14/03/16	Fri 18/03/16		Tatiana
Testes Utilizadores	9 days	Tue 22/03/16	Fri 01/04/16	26;20;21	Tatiana
Implementação	45 days	Mon 08/02/16	Fri 08/04/16		
XML linguagens + tooltip	10 days	Mon 08/02/16	Fri 19/02/16		Tatiana
Login	5 days	Mon 07/03/16	Fri 11/03/16		Tatiana
Menus	20 days	Mon 22/02/16	Fri 18/03/16	4;18;24	Tatiana

InGame	25 days	Mon 07/03/16	Fri 08/04/16		
Criar hierarquia Unity	5 days	Mon 07/03/16	Fri 11/03/16		Tatiana
GUI Logica de Jogo	15 days	Mon 21/03/16	Fri 08/04/16	5	Tatiana
Loading	15 days	Mon 14/03/16	Fri 01/04/16		Tatiana & Yuri
Networking	145 days	Mon 02/11/15	Fri 20/05/16		
Networking - criação da estrutura para utilizar frameworks de networking (abstrações)	25 days	Mon 02/11/15	Fri 04/12/15		Yuri
Networking - InGame / Logica de Jogo	25 days	Mon 18/04/16	Fri 20/05/16	32	Yuri
Networking - Lobby	5 days	Mon 02/05/16	Fri 06/05/16	32	Yuri
Networking - Chat	5 days	Mon 02/05/16	Fri 06/05/16	32	Yuri
Arte do GUI	50 days	Mon 09/05/16	Fri 15/07/16		
Desenhar icones	10 days	Mon 04/07/16	Fri 15/07/16		Tatiana
Conceito e arte final do GUI	10 days	Mon 04/07/16	Fri 15/07/16	17;23	Tatiana
Protótipo jogável v1 Milestone	0 days	Mon 09/05/16	Mon 09/05/16		
Mapa do jogo com peças	50 days	Mon 04/04/16	Mon 13/06/16		
Mapa do jogo com peças	50 days	Mon 04/04/16	Fri 10/06/16		
Construção do mapa igual ao boardgame	10 days	Mon 04/04/16	Fri 15/04/16		Yuri
Construção Mapa com editor (inclui visualização e objetivos do mapa)	10 days	Mon 04/04/16	Fri 15/04/16		Yuri
Navegação no mapa	15 days	Mon 23/05/16	Fri 10/06/16		
A*	15 days	Mon 23/05/16	Fri 10/06/16	42	Yuri
Protótipo jogável - networking + Gui + DB + mapa	0 days	Mon 13/06/16	Mon 13/06/16		
Lógica de jogo	30 days	Mon 25/04/16	Fri 03/06/16		
Combinações de Elementos	5 days	Mon 25/04/16	Fri 29/04/16		Yuri
Executar elementos (efeitos)	5 days	Mon 02/05/16	Fri 06/05/16	7;48	Yuri
Executar Abilidades	5 days	Mon 09/05/16	Fri 13/05/16	6	Yuri
Ciclo de Jogo	5 days	Mon 16/05/16	Fri 20/05/16		Yuri
Interação entre peças e mapa	10 days	Mon 23/05/16	Fri 03/06/16	42	
Implementação	10 days	Mon 23/05/16	Fri 03/06/16		Yuri
GUI	10 days	Mon 23/05/16	Fri 03/06/16		Tatiana
Art	6 days	Mon 11/04/16	Mon 18/04/16		
Art - Mundo	3 days	Mon 11/04/16	Wed 13/04/16		Tatiana

Art - Herois	3 days	Thu 14/04/16	Mon 18/04/16		Tatiana
Projecto completo - falta 3D e IA	0 days	Mon 06/06/16	Mon 06/06/16		
3D	59 days	Tue 19/04/16	Fri 08/07/16		
Modelação	29 days	Tue 19/04/16	Fri 27/05/16	57	Tatiana
Texturização	5 days	Mon 30/05/16	Fri 03/06/16	60	Tatiana
Rigging	10 days	Mon 06/06/16	Fri 17/06/16	61	Tatiana
Animação	15 days	Mon 20/06/16	Fri 08/07/16	62	Tatiana
Particulas de tudo	10 days	Mon 11/07/16	Fri 22/07/16	27;59	Tatiana
Playtesting - 1	5 days	Mon 13/06/16	Fri 17/06/16	58	Yuri;Tatiana
Inteligencia artificial	30 days	Mon 13/06/16	Fri 22/07/16		Yuri
Play					
Basic Strategy					
Outras coisas					
Projeto completo Milestone	0 days	Fri 22/07/16	Fri 22/07/16		
Escrita da Tese - Tati	15 days	Mon 25/07/16	Fri 12/08/16	64;59;36	Tatiana
Escrita da Tese - Yuri	20 days	Mon 25/07/16	Fri 19/08/16		Yuri
Tese concluída					

9.2 Requirements

9.2.1 Game Abilities

Table 7 – Full table of abilities with their codenames, ingame names, ingame description, range, cost and indication if it is a cycle or not. There is also their ID in the database, located below the codename.

CodeName	Name	Description	Range	Cost	Cycle
Ghora1 1	Ghora 1	Ghora deals 1 damage and an additional 1 damage for every 2 health he has lost.	1	1RC	
Ghora2 2	Ghora 2	Ghora deals 1 damage and an additional 1 damage for every 3 health he currently has.	1	1RC	
Ghora3 3	Ghora 3	Ghora jumps up to 3 tiles and deals 1 damage to all enemies around the landing tile.	-	1RC	
Ghora4 4	Ghora 4	Redirects to Ghora an ability that would affect a friendly character next to him.	1	1RC	X
Ghora5 5	Ghora 5	[COUNTER] Ghora reflects the damage of an ability targeting him. Every other effect is cancelled.	-	1RC	

Ghora6 6	Ghora 6	Ghora deals 2 damage to all enemies and 2 healing to all allies within 1 tile.	-	1RC	
GhoraU1 7	Ghora U1	Ghora becomes immortal. Damage from abilities or ultras that would kill him, drains resources instead of health.	-	2RC	X
GhoraU2 8	Ghora U2	[COUNTER] Ghora takes the damage that would hit an ally and heals them by half the damage that Ghora takes.	1 to 4	2RC	
KULLA					
Kulla1 9	Kulla 1	Kulla deals 2 damage and if the target dies, all characters of the same team within 1 tile will take 1 damage	2 to 3	1RC	
Kulla2 10	Kulla 2	[COUNTER] Kulla cancels the next damaging ability that would hit her	-	1RC	
Kulla3 11	Kulla 3	Choose one: Kulla shoots an explosive trap that immediately deals 1 damage to all characters around the target tile. Kulla arms 2 traps on the map that deal 2 damage once triggered. These traps cannot be placed on tiles next to or with a character on them.	2 to 3	1RC	X
Kulla4 12	Kulla 4	Kulla deals 2 damage and moves the target 1 tile closer to her.	4	1RC	
Kulla5 13	Kulla 5	Kulla rolls over 2 tiles and applies her element to all characters on the tiles around her landing.	-	1RC	
Kulla6 14	Kulla 6	Kulla deals 1 damage to the target. If there is another character of the same team next to the target, Kulla can incapacitate it for one turn	2 to 3	1RC	
KullaU1 15	Kulla U1	Kulla deals 3 damage. Spending 1 additional resource will increase the range of this ability by 1 tile, up to a maximum of 3	2 to 3	2RC	
KullaU2 16	Kulla U2	Kulla deals 3 damage to the target closest to her and 2 damage to the target furthest from her.	2 to 3	2RC	
LIPP					
Lipp1 17	Lipp 1	[COUNTER] Lipp cancels an ability targeting her or an ally. If Lipp is the target of the ability, she deals 1 damage to the attacker.	1 -	1RC	
Lipp2 18	Lipp 2	Choose one: Lipp switches places with her target and then pushes every character within a 2-tile radius by 2 tiles Lipp switches places with her target and then pulls every character within a 2-tile radius by 2 tiles	1 to 2	1RC	

Lipp3 19	Lipp 3	Lipp deals 3 damage to the target. If the target dies, Lipp deals an additional 2 damage to another target of the same team within 2 tiles.	1 to 2	1RC	
Lipp4 19	Lipp 4	Lipp deals 1 damage to the target. If the target dies, Lipp deals an additional 4 damage to another target of the same team within 2 tiles.	1 to 2	1RC	
Lipp5 20	Lipp 5	Lipp deals 2 damage to the target and an additional 2 damage to another target of the same team within 2 tiles.	1 to 2	1RC	
Lipp6 21	Lipp 6	Once activated, every time Lipp takes damage, the attacker takes 1 damage in addition to the damage dealt to Lipp. Spends 1 resource for each damage returned.	-	1RC	X
LippU1 29	Lipp U1	Lipp reverses the state of all her abilities. Used abilities become unused and unused abilities become used.	-	2RC	
LippU2 23	Lipp U2	<p>Copies an Ultra from the target. Lipp may copy another ultra to change the currently active ultra</p> <p>Ghora: GhoraU1 - 24 {7}</p> <p>Kulla: KullaU2 - 25 {16}</p> <p>Loper: LoperU2 - 26 {37}</p> <p>Malik: MalikU2 - 27 {45}</p> <p>Wang: WangU1 - 28 {52}</p>	1 to 5	2RC	
LOPER					
Loper1 30	Loper 1	Loper exchanges his health with the targets' current health. Neither Loper nor the target can exceed their own maximum health.	1 to 2	1RC	
Loper2 31	Loper 2	[COUNTER] Loper can sacrifice himself and die instead of an ally within range. The ally stays alive with Loper's current health. Cannot exceed the targets' maximum health.	1 to 5	1RC	
Loper3 32	Loper 3	Loper deals 2 damage and is healed for 2 health.	1 to 2	1RC	
Loper4 33	Loper 4	Loper deals 2 damage and if the target dies, Loper can use one of its abilities.	1 to 2	1RC	
Loper5 34	Loper 5	Loper transfers 1 health from all heroes 2 tiles away from the target to the target.	1 to 2	1RC	
Loper6 35	Loper 6	The damage taken by an ally character in a tile next to Loper is equality distributed between the two. Damage is rounded down.	0	1RC	X

LoperU1 36	Loper U1	Loper resurrects an ally and places them on a tile next to him.; Loper resurrects an ally and places them on a tile next to him. Loper sacrifices 4 health instead of paying resources.	-	2RC	
LoperU2 37	Loper U2	Lopper resurrects an ally under his control for 1 turn. The ally will resurrect on a tile next to Lopper and for each resource they would spend, Lopper takes 1 damage instead. The ability will wear off if Lopper would be killed by this damage.	-	2RC	
MALIK					
Malik1 38	Malik 1	Malik hides in the shadows and can't use Damage Abilities while hidden. Malik can only be targeted by element powers and Utility abilities cost 1 more resource. When Malik leaves the shadows, he deals 2 unavoidable damage to his target.	- / 1	1RC	X
Malik2 39	Malik 2	Malik deals 2 Damage. If he doesn't have Malik 1, he deals 1 additional damage.	1	1RC	
Malik3 40	Malik 3	Malik can move 1 tile to reach the target and deals 1 damage. Malik deals 2 damage if he doesn't need to move	1 to 2	1RC	
Malik4 41	Malik 4	Choose one: Malik picks the target's pockets and places one of their damage abilities on cooldown.; Malik protects himself from an ability during the enemy turn.	1	1RC	
Malik5 42	Malik 5	Choose one: Malik removes a trap within range.; Restore 1 health point to Malik.	1	1RC	
Malik6 43	Malik 6	Malik creates a smoke cloud around him that prevents all characters from using or being targeted by abilities.	-	1RC	X
MalikU1 44	Malik U1	Malik deals 3 damage and an additional 1 damage for each character within 2 tiles of him.	1	2RC	
MalikU2 45	Malik U2	Malik prepares to instantly kill an opponent, immediately ending his turn. During the next turn, Malik can instantly kill any target within range. This ability will reset automatically even if no target is killed.	1	1RC / 2RC	
WANG					
Wang1 46	Wang 1	Wang deals 2 damage on 3 tiles around to him.	1	1RC	
Wang2 47	Wang 2	Wang uses his chains to deal 1 damage and moves the target up to 2 tiles.	2 to 3	1RC	

Wang3 48	Wang 3	Wang deals 2 damage and if the targets dies, Wang is healed by 2.	1	1RC	
Wang4 49	Wang 4	Wang moves to the closest tile next to the target and deals 2 damage. If there is another target of the same team within 2 tiles of the first, Wang will move to the closest tile next them and deal 1 damage.	2	1RC	
Wang5 50	Wang 5	[COUNTER] Wang redirects the damage from an ability that would hit him to another target. All secondary effects are cancelled.	1 to 2	1RC	
Wang6 51	Wang 6	Wang changes the position of two characters. Both characters must be within Wang's range.	1 to 2	1RC	
WangU1 52	Wang U1	Wang deals 2 damage and an additional 2 damage if the target has less than half of its maximum health	1	2RC	
WangU2 53	Wang U2	Wang creates a clone of himself. Wang and the clone move separately but share abilities and their cooldowns. As soon as the clone takes damage it disappears.	1 to 3	2RC	X

9.2.2 Element Powers

Table 8 – Full table of Element Powers with their name, description, time to use and rarity. Rarity indicates how many of that card exists in the game: 3 common, 2 rare and 1 epic.

Name	Description	Time to Use	Rarity
F1	Deal 1 damage to the last character of the opposing team that played and 1 damage to another one.	Instant	Common
F2	Deal 1 damage to the last character of the opposing team that played and 2 damage to another one.	Instant	Rare
F3	Deal 1 damage to the last character of the opposing team that played and 3 damage to another one.	Instant	Epic
F4	Deal 1 damage for each utility ability the character currently playing has. Deals at least 1 damage.	Instant	Rare
F5	Deal 3 damage plus, 2 for each character of the same team in an adjacent tile to the target.	Instant	Epic
F6	Choose one: Draw a Water elemental power Draw a Lightning elemental power Deal 1 damage to a character.	Instant	Common
F7	Choose one: Draw a Wind elemental power. Draw an Earth elemental power. Deal 1 damage to a character.	Instant	Common
F8	Deal 1 damage to a character for each character controlling Fire in the game. Deals at least 1 damage.	Instant	Common
F9	Choose one: Restore 1 resource for each damage ability your team has used. Deal 1 damage to a character.	Instant	Common

F10	Deal 1 damage plus, 1 for each additional ability the opposing team has unlocked.	Instant	Rare
F11	Deal 1 damage to all characters of the opposing team. Deals 1 more if one of your bases has been destroyed.	Instant	Rare
WATER			
A1	Heal 1 health to the last character of your team that played and 1 health to another one.	Instant	Common
A2	Heal 1 health to the last character of your team that played and 2 health to another one.	Instant	Rare
A3	Heal 1 health to the last character of your team that played and 3 health to another one.	Instant	Epic
A4	Heal 1 health for each damage ability the character currently playing has. Heals for at least 1 health.	Instant	Rare
A5	Heal 3 health plus, 1 health for each character of the same team in an adjacent tile to the target.	Instant	Rare
A6	Choose one: Draw a Fire elemental power. Draw a Wind elemental power. Heal 1 Health to a character.	Instant	Common
A7	Choose one: Draw a Lightning elemental power Draw an Earth elemental power Heal 1 health to a character.	Instant	Common
A8	Heal 1 health to a character for each character controlling Water in the game. Heals for at least 1 health.	Instant	Common
A9	Heal 1 health to all characters of your team for each ability of the opponent team has and 2 health for each unlocked ultra	Instant	Epic
A10	Heal 1 health plus, 1 health for each additional ability the opposing team has unlocked	Instant	Rare
A11	Heal 1 health to all characters of your team. Heal 1 more health to each character if one of your bases has been destroyed.	Instant	Common
WIND			
V1	Heroes from the opposing team that are inside the Base area of this team will suffer damage equal to the Base's damage. At the start of the round every hero from the opposing team is moved outside of the Base area and if, they re-enter it during this round they will take damage equal to the Base's Damage.	Element Zone	Epic
V2	Heroes from the opposing team are moved outside of this team's Base areas and are pushed away by 1 + Base's damage tiles.	Save for later	Rare
V3	You can move 2 heroes for up to 3 tiles each.	Instant	Rare
V4	After being damaged by an ability, a hero can move as many tiles as the damage taken	Element Zone	Rare
V5	Choose one: Draw a Water elemental power. Draw an Earth elemental power. Heal a Base by 1 health. Move 1 of your team's heroes by up to 3 tiles.	Instant	Common

V6	Choose one: Draw a Lightning elemental power. Draw a Fire elemental power. Heal a Base by 1 health. Move 1 of your team's heroes by up to 3 tiles.	Instant	Common
V7	Choose one: Move 1 of your team's heroes up to 3 tiles for each hero that controls Wind in the game. Heal a Base by 1 health for each hero that controls Wind in the game.	Instant	Common
V8	During this round, heroes from the opposing team can't stay inside this team's Base areas while those Bases are alive. However, they can use these areas to move. If any heroes remain inside the area after their movement, they will be moved to the closest tile outside the of the area.	Element Zone	Common
V9	The hero playing can move up to 6 tiles during its first movement phase, instead of 3.	Save for later	Common
V10	Heroes from this team can't be targeted by abilities from the opposing team while standing in their own Base area.	Element Zone	Rare
V11	Move one hero up to 12 tiles. If you choose to move a hero from the opposing team, they must be moved to the centre of the map. You can move 1 additional hero if one of your Bases has been destroyed.	Save for later	Epic
EARTH			
T1	The heroes from the team that uses this power deal 1 more damage to Bases but, also take 1 more damage.	Element Zone	Rare
T2	Deal 2 damage to a Base if there are two heroes from your team inside that Base area.	Save for later	Common
T3	Choose one: After killing a Base, fully heal the hero that killed it. One of your heroes takes no damage when attacking a Base.	Save for later	Rare
T4	When attacking a Base, you can deal 2 more damage to it.	Save for later	Epic
T5	Choose one: Draw a Lightning elemental power. Draw a Wind elemental power. Deal 1 damage to a Base if your team has at least 1 hero inside its area.	Instant	Common
T6	Choose one: Draw a Water elemental power. Draw a Fire elemental power. Deal 1 damage to a Base if your team has at least 1 hero inside its area.	Instant	Common
T7	If the hero playing is inside an opposing Base area, he can deal 1 more damage for each hero controlling Earth in the game.	Save for later	Common
T8	If the hero currently playing attacks a Base, they will deal 1 more damage and take 1 less damage.	Save for later	Rare
T9	During 1 round, the heroes from your team are immune to Base damage.	Element Zone	Common
T10	If your heroes are inside the opposing Base area, they can't be targeted by enemy abilities.	Element Zone	Rare
T11	The hero currently playing can teleport to a Base area and deal 1 damage to it.	Save for later	Epic
LIGHTNING			
R1	You can use an ability for free, without paying any cost or activating its cooldown.	Save for later	Epic
R2	Inverts the effect of the ability being played. Healing now damages and damage now heals	Save for later	Common

R3	Steals an elemental power from the opposing team.	Save for later	Common
R4	Reduces the cost of one ultra by 2 resources.	Save for later	Rare
R5	Forces the opposing team to swap the last unlocked ability of the targeted hero.	Instant	Rare
R6	During one round both teams can't use more abilities than the team that has the least abilities.	Element Zone	Rare
R7	Choose one: Draw an Earth elemental power Draw a Fire elemental power Deal 1 damage to a Hero Store and use it when inside a Base area to deal 1 extra damage.	Instant	Common
R8	Choose one: Draw a Water elemental power Draw a Wind elemental power Heal 1 health to a Hero Move a Hero up to 3 tiles.	Instant	Common
R9	Choose one: Deal 1 damage to a character and to a Base. Heal 1 health to a character and to a Base.	Instant	Common
R10	Increases the cost of the opposing team's abilities by 1 Resource. If you have unlocked less abilities than the opposing team, the cost is increased by 1 more resource.	Element Zone	Rare
R11	Unlocks 1 ability for every 3 abilities the opposing team has.	Instant	Epic

9.3 Game Systems

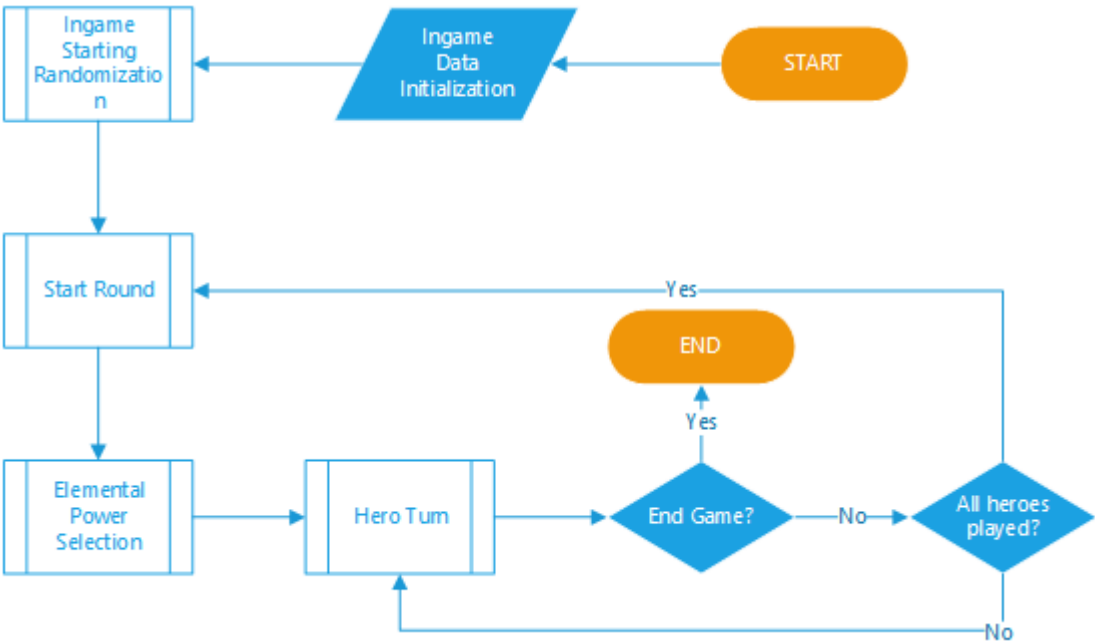


Figure 43 – Game Cycle flow chart.

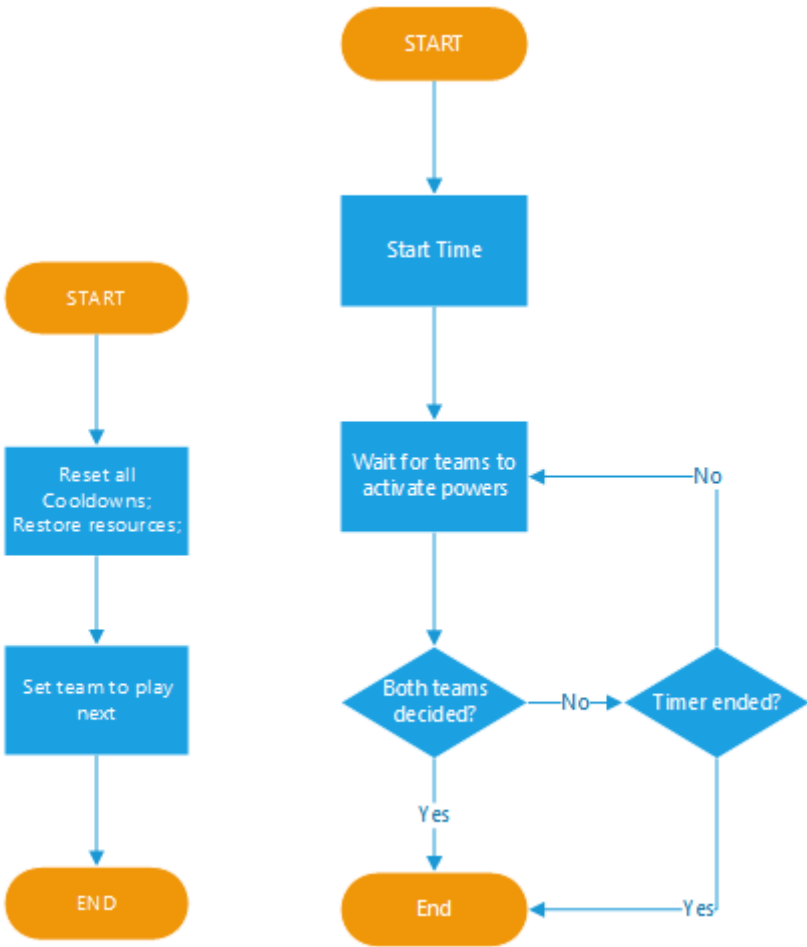


Figure 44 – New round and Choose Elemental Power Zone element flowcharts.

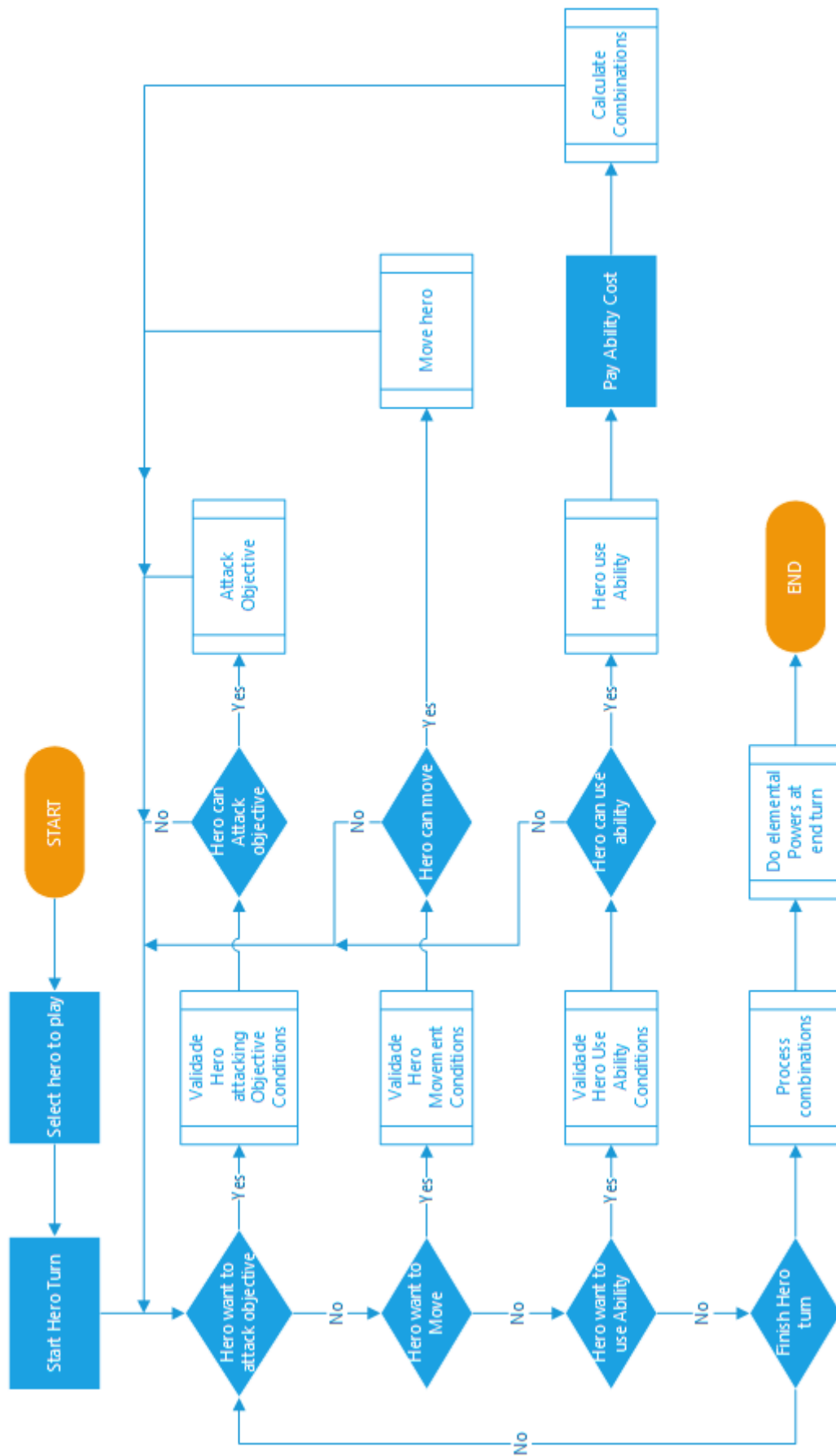


Figure 45 – Hero turn flowchart

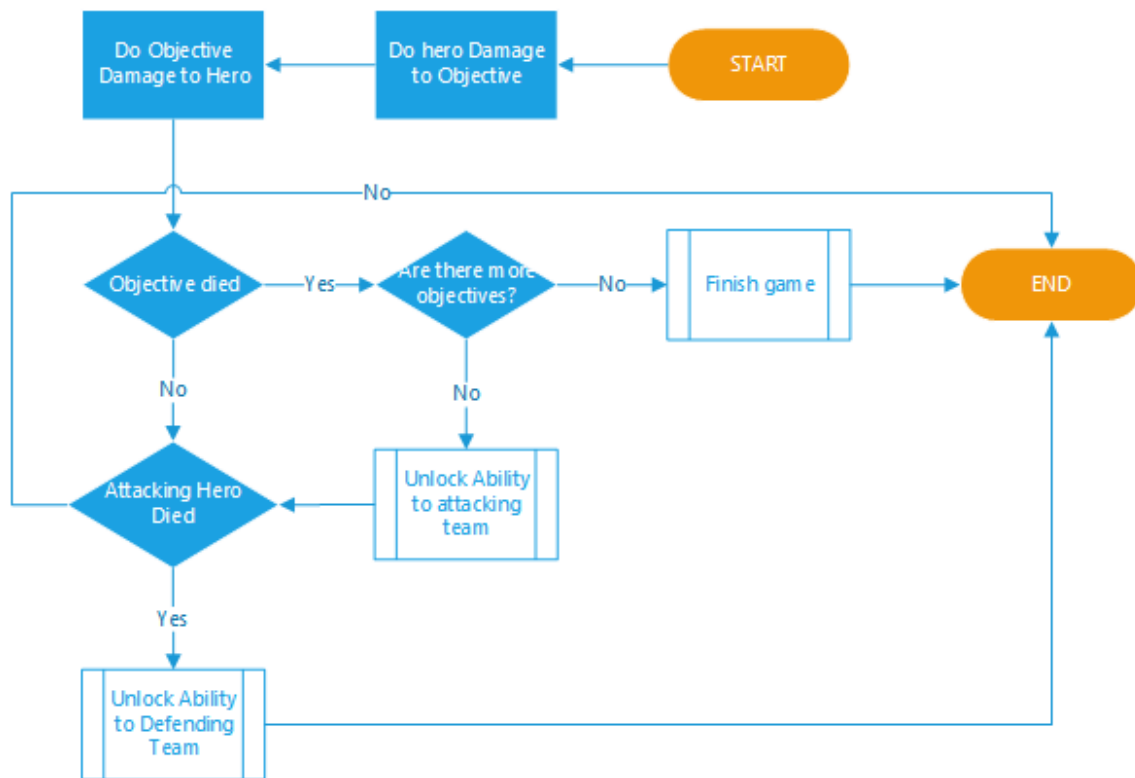


Figure 46 – Hero attack Objective flowchart

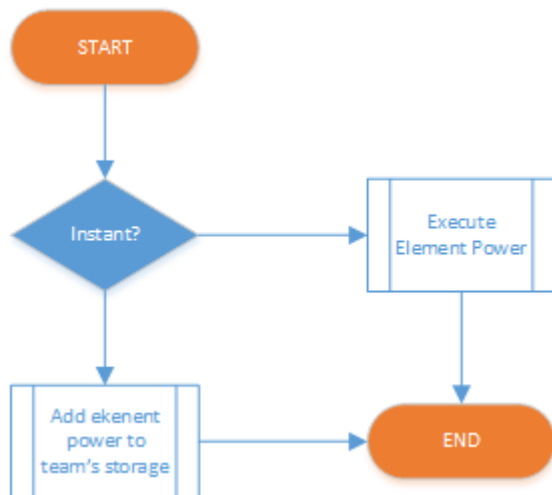


Figure 47 – Gain Element Power flowchart

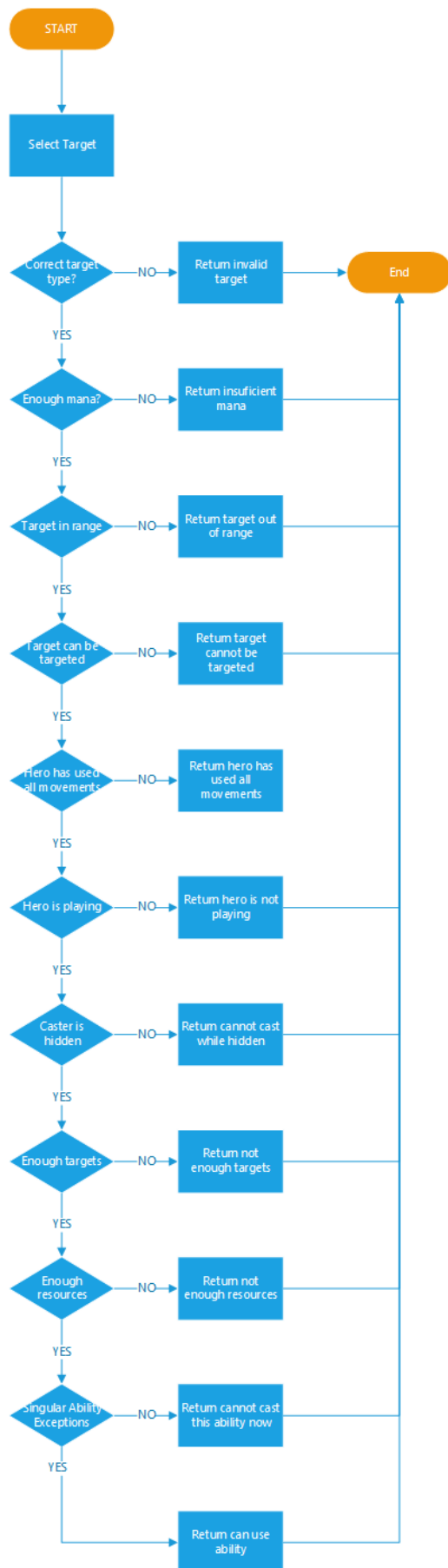


Figure 48 – Verify can execute ability flowchart

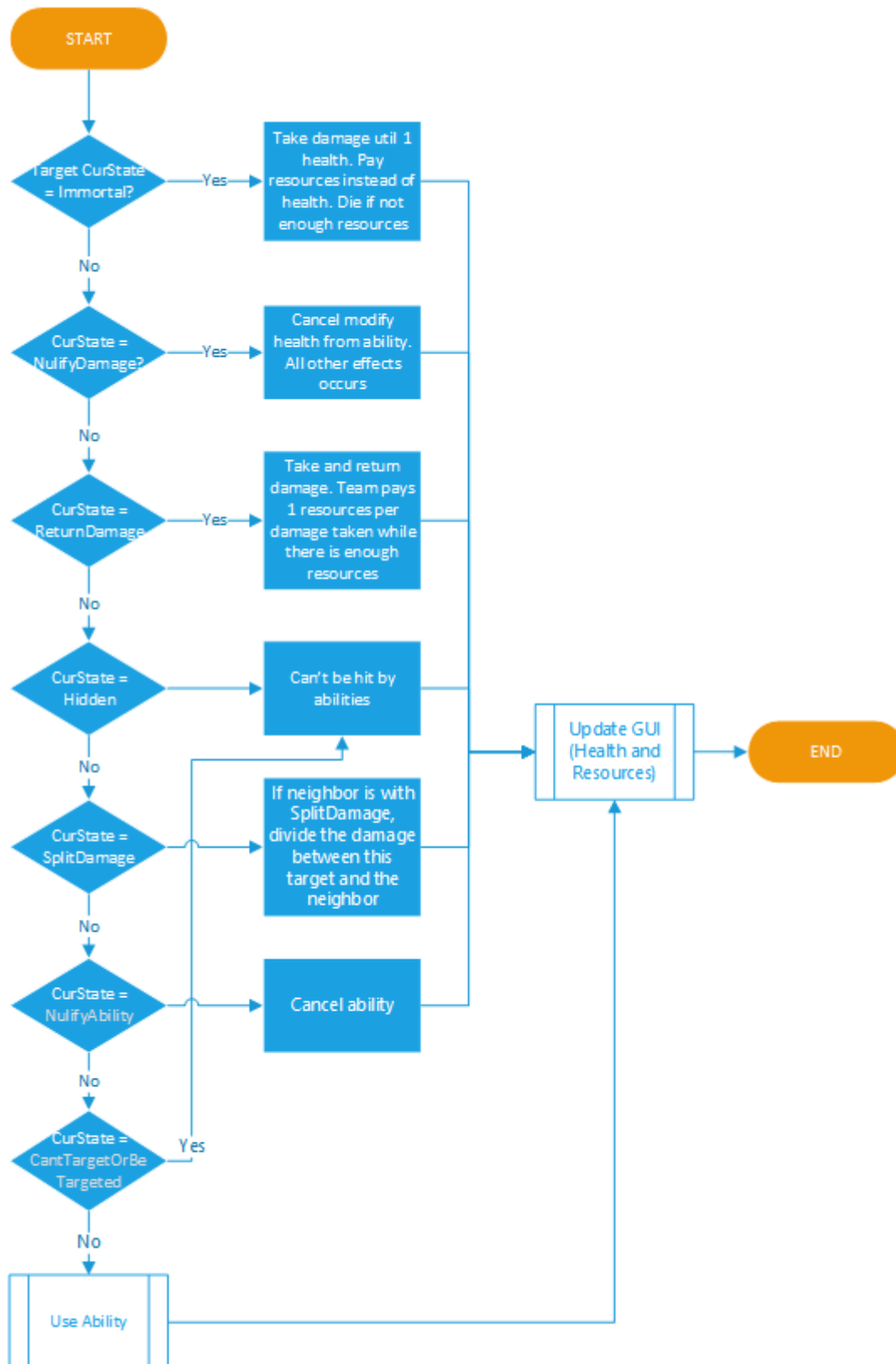


Figure 49 – Execute Ability flowchart

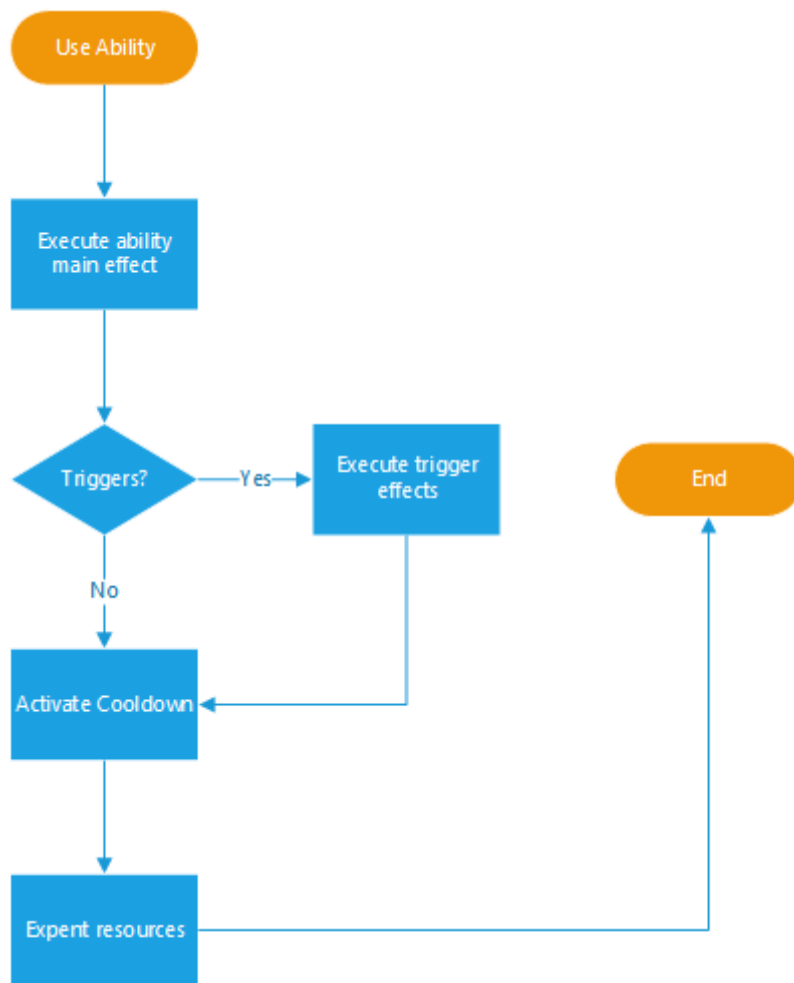


Figure 50 – Use ability flowchart

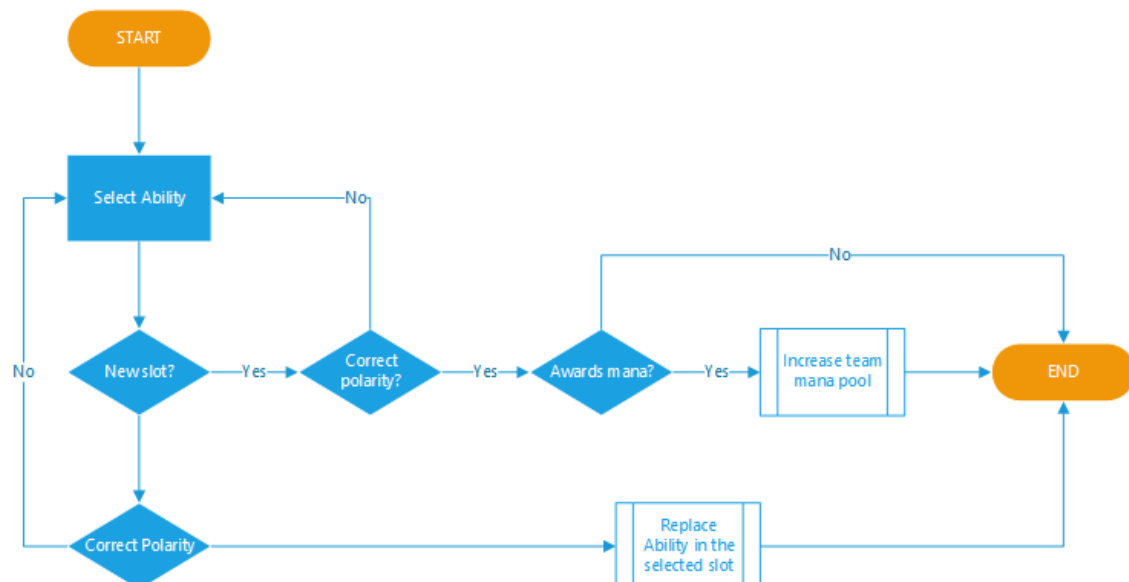


Figure 51 – Unlock Ability flowchart

9.4 Database

9.4.1 Account tables

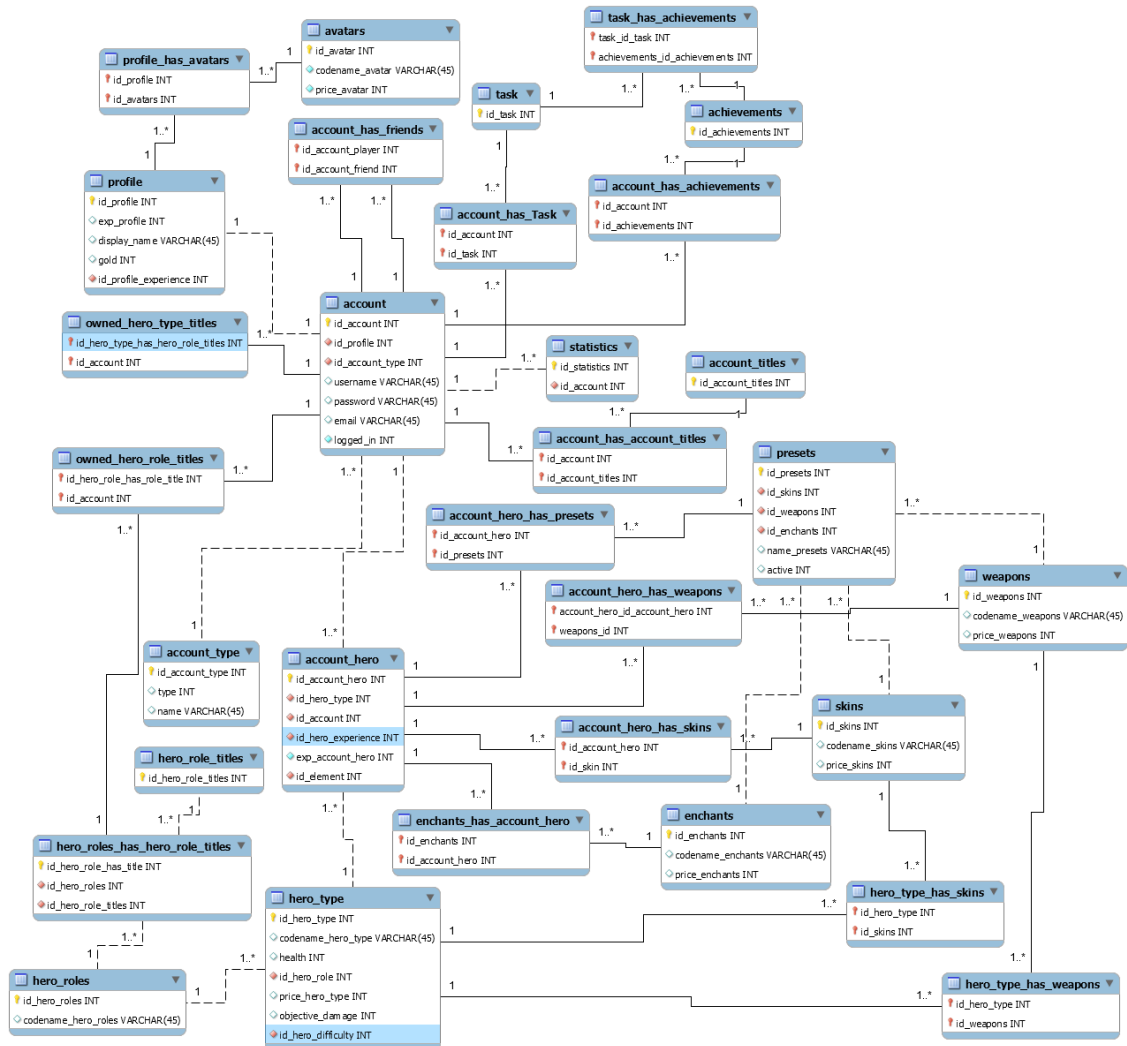


Figure 52 – Account tables with all possessions and achievements.

9.4.2 Elements balancing table

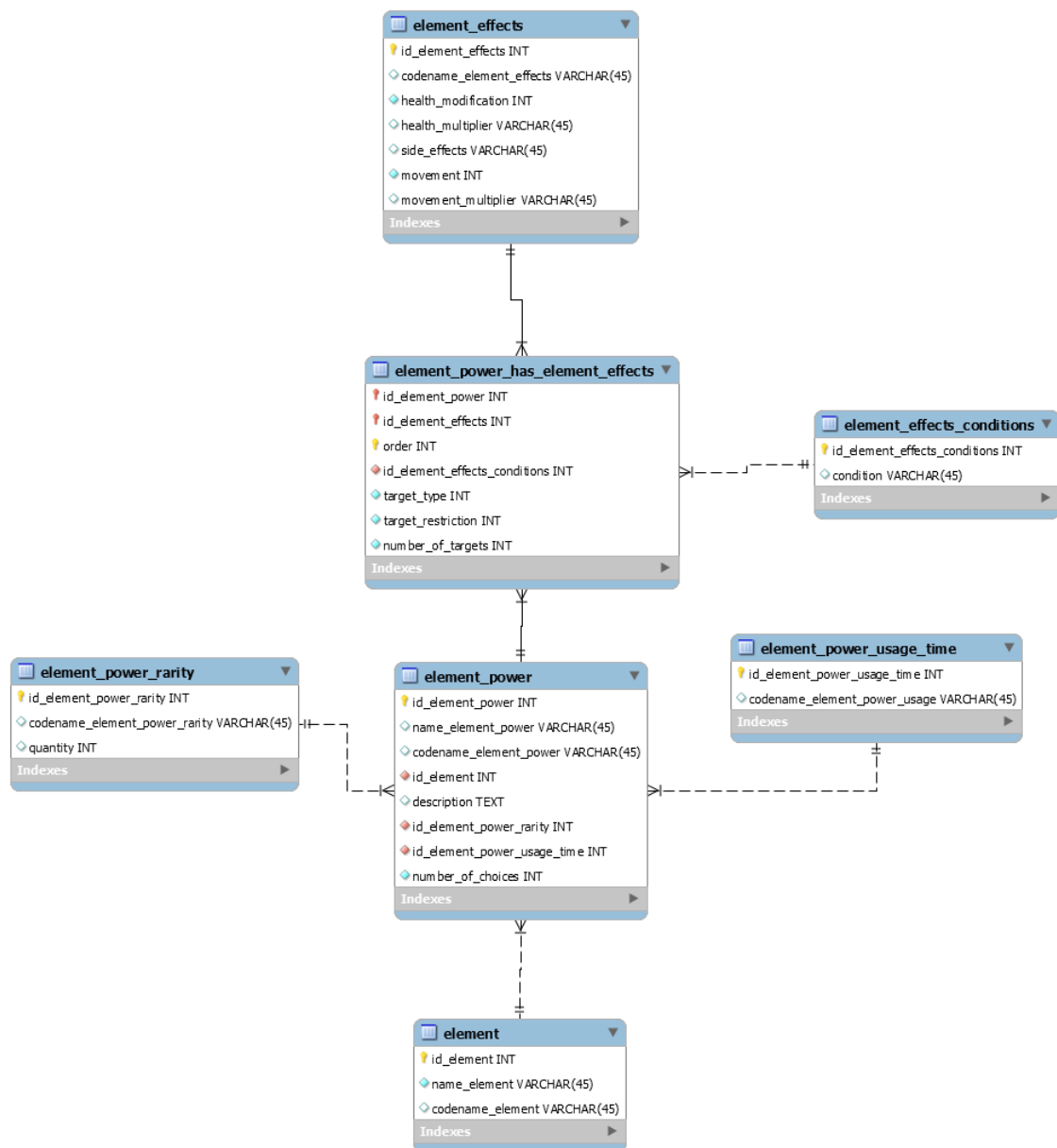


Figure 53 – Elements and Element Power balancing tables

9.4.3 Heroes and Abilities balancing tables

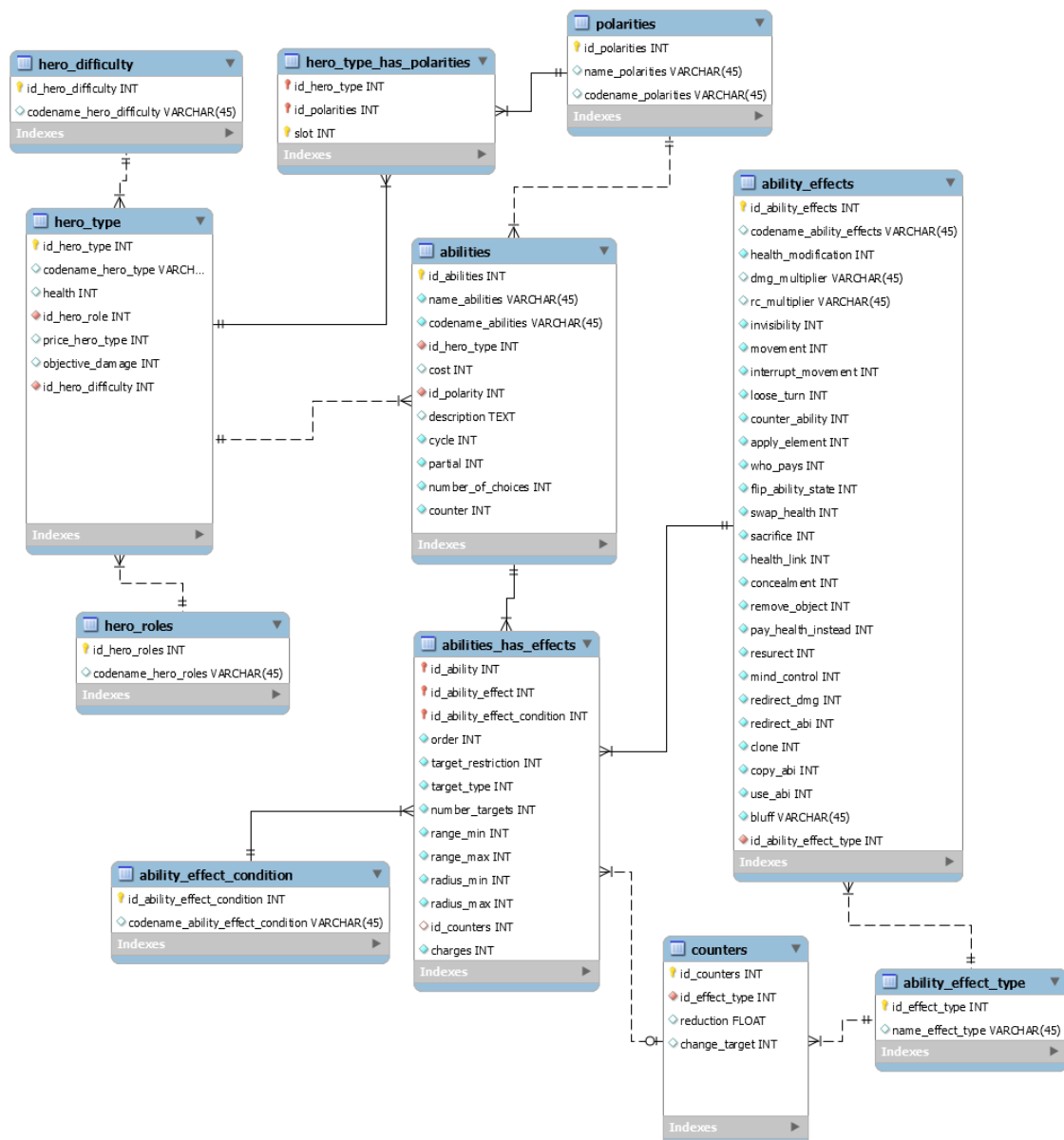


Figure 54 – Heroes and Abilities balancing tables.

9.5 Bug reports

9.5.1 Playtest 23-11-2016

1. Falta adicionar cooldown na peça na primeira ronda depois de fazer spawn
2. Foi feita uma combinação neutra de raios no turno 1 ronda 1(malik ligou o stealth) e sai raios R9 para a equipa do malik e depois saiu R5 para a outra equipa.. Quando acabou de resolver a R5 mudou o turno para a mesma equipa outra vez
3. Quando o juan (equipa do malik) carregou end turn deu um out of range exception e nao era possivel nenhuma equipa começar um novo turno
4. Depois de usar a carta F10 deu um ArgumentOutOfRangeException (comment)
5. Depois de usar a F9 o jogo nao passou para a outra equipa começar o turno

6. Fazer res com um heroi nao esta a por o elemento dele.. Ta a ficar com o que tinha quando morreu
7. Andei com a lipp e deu um erro



8. Saiu a carta V5, eu escolhi mover uma peça, escolhi a peça, mudei de ideias e carreguei noutra peça e fechou o target's panel se nao pude acabar de resolver o elemento
9. Mudei de turno usando o cheat panel para poder saltar para poder jogar mas quando tentei usar as habilidades deu erro InvalidOperationException (que pelo nome faz sentido pk o elemento nunca foi resolvido logo nao posso usar as habilidades)
10. A char que leva com o stun da kulla fica enfiada dentro da terra
11. Numa combinaçao neutra, se o primeiro mata (E desboqueia uma habilidade), a combinaçao do segundo não é resolvida
12. [maybeBug] Eu fui fazer res com a minha lipp e saltou automaticamente para o turno do juan e eu nao fiz spawn
13. Lipp andou e deu um OutOfRange



14. Quando se faz end turn com o targets panel aberto esconde o painel mas o botao do use fica a aparecer

15. A T10 esta bugada, protege os herois na sua zona dastorres e na do adversario



16. Fazer reflect (com o ghora) do malik 2 refletiu o dano mas levei dano na mesma

17. Acho que o que aconteceu foi refletiu a primeira parte da habilidade (2 de dano) mas nao a segunda parte (+1 de dano se nao tiver o stealth)

18. O malik do juan fez uma combinação negativa Raios para o meu ghora (fogo acho eu) e a interface disse que tinha ganho carta de raios mas levantou uma de fogo (F7)

19. O ultimo heroi que jogou fez res e o jogo nao fez new round

20. O malik matou um heroi, desbloqueou o Malik U2 e nao o consegue usar outras habilidades (testado o Malik U2 e malik5 - cura) - n consegui reproduzir

21. [Sugestao] quando se gasta

22. É possível fazer discard as cartas guardadas
23. Fazer discard reduz o contador de cartas mas não reduz o número de bolinhas dos elementos
24. O reflect do ghora está a reflectir o dano MAS: o ghora continua a levar o dano na mesma e não cancela os outros efeitos/triggers da habilidade
25. A kulla usou o Kulla 6 no ghora (dano) e no malik (Stun) e o ghora levou dano, o malik levou dano e a kulla levou com o dano de volta
26. Sincronização da teamcomp no início do jogo está a perder a informação sobre os elementos (se um muda os elementos, o outro vê o que estava na base de dados antes de mudar o elemento)



27. New round n faz reset ao tempo
28. O timer quando chega a 30 segundos acrescenta 1 minuto
29. Tempo total em vez de ronda
30. Partículas do Malik foram-se
31. Não faz update ao gui depois de andar 2 vezes (falta trancar o ataque da torre)
32. O update da gui tem de ser feito olhando para quantas vezes ele já andou neste lugar específico

33. Malik escondido nao pode ser afetado por habilidades (incluindo AOE e triggers)
34. V8 da pa usar sem estar na altura da ZE
35. [GUI] Pedrinha no MalikStoneTrap ta descendo - 29/11/2016
36. [GUI] Malik1 anim muito rapida - 29/11/2016
37. Grid de andar - é preciso ver se é da equipa do jogador - 29/11/2016
38. Tirar self da lista de targetables
39. Passa duas rondas no fim da primeira (spawns)
40. Remover trap tem de fazer combinação de elementos
41. [GUI] Ghora lobo ta desincronizado acontece mt cedo - 29/11/2016

9.5.2 Playtest 28-11-2016

1. O vento fica fora do ecra para a equipa armatheus quando se joga com a chronis - 29/11/2016
2. [TODO] Aparece o contador da zona dos elementos quando so se tem cartas para usar mais tarde
3. Pode aparecer uma barra vazia a dizer que nao ha zona dos elementos
4. 16:58 bronca nas combinações (Video 2016-11-28 16-45-03 Joao)
5. Não mudou de equipa no segundo que recebeu elemento de uma combinação neutra. O primeiro usou cartas de tirar outras cartas
6. Quando pus as traps da kulla (opção 2) gastou a segunda fase de movimento
7. Ou seja nao me deixou atacar a torre 8 o problema nao foi gastar o movimento)
8. A lipp gastou as duas fases de movimento mas pode ativar a lipp6
9. A lipp pode desativar o lipp6 no turno em que ativa
10. [GUI] No V5 quando se escolhe a opção 4 (mover 1 peça) se tivermos um heroi como alvo e carregarmos com o botao esquerdo noutra heroi para o selecionar em vez do que tinhamos o targets panel desaparece Video (2016-11-28 17-18-12 Juan @ 03:59)- 29/11/2016
11. [NAO SABEMOS O QUE ACONTECEU!!!!] Saiu a carta F9 ao Juan e ele foi corrido do jogo (Video 2016-11-28 17-28-09 Juan @ 02:50)
12. Provavelmente levou DC
13. A carta L9 nao deixa limpar a torre do alvo secundário dá um erro "invalid Operation Exception" Video (2016-11-28 17-34-10 Joao @ 15:12) - 29/11/2016
14. As traps da kulla opção 1 nao afetaram o malik quando ele tava numa casa encostada e nao estava escondido Video 2016-11-28 17-54-15 Joao @ 08:54)
15. O problema era o Malik 3 não ocupava o novo hex depois de se mover... Tati ve se o que ue fiz está bem sff


```

/// <summary>
///
/// </summary>
1reference
private IEnumerator ExecuteAbi3Anim(AbilityParameters parameters, AnimAttack attackAnim, AbilityNumber abiNumber)
{
    var caster = parameters.Caster;
    var target = parameters.HeroTargets[0];
    var abi = parameters.IngameAbility;
    var dist = MapMath.DistanceInHexes(caster.MonoHex, target.MonoHex);

    if (dist == 2)
    {
        //caster.Move(parameters.TriggerHexTargets[0]);
        var targetHex = parameters.TriggerHexTargets[0];
        StartCoroutine(MoveBeforeAttack(targetHex));

        ExecuteAbilityGui.Instance.MoveHeroWithAbility(caster, parameters.TriggerHexTargets[0], true);
        caster.MonoHex.OccupyHex(caster, true);
        //andar

        while (!canAttack)
        {
            yield return new WaitForSeconds(0.01f);
        }
    }

    MonoHero.LookTo(target);
    Attack(attackAnim, abiNumber);
}

```

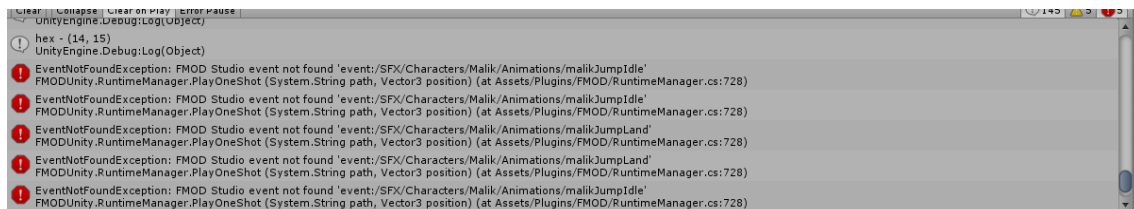
16. O malik nao estava escondido e nao podia ser target de habilidades Ver SS abaixo e Video 2016-11-28 17-54-15 Joao @ 11:41)



17. Não dá para usar a segunda opção do Malik4 (Video 2016-11-29 17-17-50 Juan @ 04:12) - Nao preenche o target
18. [GUI] Ficou a seta de target em cima do ghora do joao depois do targets panel ter sido fechado pelo end turn (Video 2016-11-29 17-17-50 Juan @ 06:30) - 29/11/2016
19. Nao estava a calcular com deve ser os movimentos possiveis da lipp (SS abaixo) (Video 2016-11-29 17-17-50 Juan @ 10:30) VEJAM O VIDEO!!! Muito estranho



20. O ghora do Juan foi selecionado e o movimento possível foi bem calculado (Video 2016-11-29 17- Juan @ 10:49)
21. A lipp estava selecionada mas estava a contar como se fosse a Kulla
22. (Video 2016-11-29 17- Juan @ 13:48)
23. A kulla nao podia ser target de habilidades (Video 2016-11-29 17-17-50 Juan @ 15:45)
24. VER SE A LIPP USOU O TRADE - provavelmente a mudança dos hexes ainda nao estao direitos
25. Nop, nao é :(- desisto _ _;
26. Yuri - dá erro no fmod quando uso o lipp2, não consigo testar



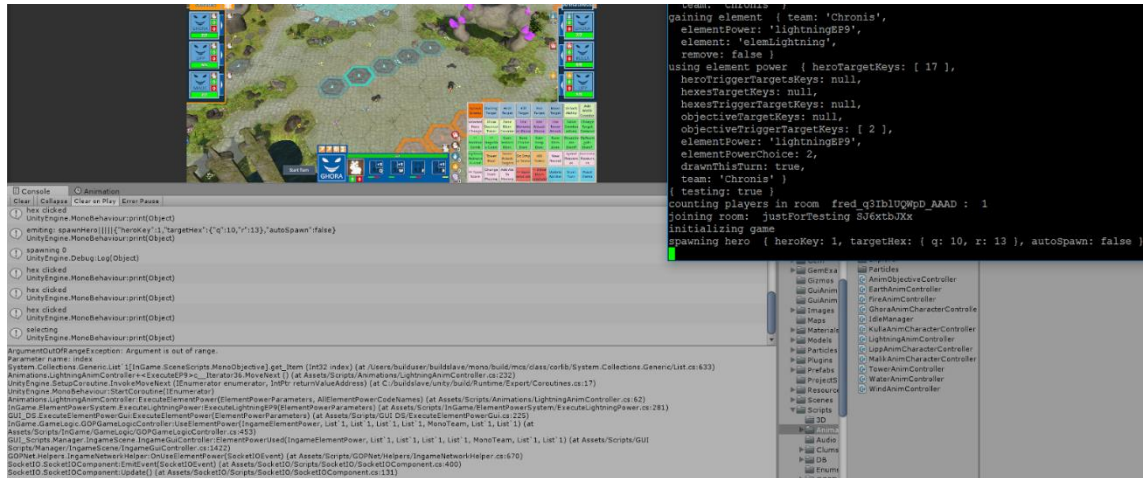
27. Ja implementaram os turnos com tempo? Acabou o turno do meu ghora e ainda tinha ações por fazer (Video 2016-11-29 17-48-29 Joao@ 13:15)
28. O botao do start turn ficou bugado no do end turn (Video 2016-11-29 17-48-29 Joao@ 14:51)
29. A lipp do Juan estava imune a habilidades:
30. Leva rez a (Video 2016-11-29 17-48-28 Juan @ 12:31)
31. Resolve a ultima combinação de elementos da ronda Video 2016-11-29 17-48-28 Juan @ 14:45
32. Muda de ronda a tem e a lipp ganha o buff "CanNotbetargeted" Video 2016-11-29 17-48-28 Juan @ 14:46
33. Desapareceu quando ela saiu da zona das torres (Video 2016-11-29 17- Juan @ 33:50)
34. A carta F4 foi usada na lipp que tinha 2 habilidades de utilidade mas so fez 1 de dano (Video 2016-11-29 17-48-29 Joao@ 26:40)

35. Está a funcionar bem. O ultimo heroi a jogar foi o Malik e ele não tem nenhuma habilidade de utilidade.



36. [GUI] O targets panel nao muda de atacar a torre para usar carta do elemento (Video 2016-11-29 17-48-29 Joao@ 25:24) - 29/11/2016
37. Nem para habilidades Video 2016-11-29 17-48-29 Joao@ 35:37
38. Nao foi possivel atacar a torre do juan depois de sair de stealth com o malik (Video 2016-11-29 17-48-29 Joao@ 24:54) - Yuri doing overtime from home! :D .. sry não resisti, esta chamou-me muito a atenção e tive de ver o que se passava... afinal não era possivel atacar depois de usar qualquer habilidade :D já ta corrigido ;)
39. [Aconteceu qualquer coisa e o dano das torres nao tava sincronizado]
40. O joao ataca a torre do Juan com a T5 2 vezes: Video 2016-11-29 17-48-29 Joao@ 25:38 (Video 2016-11-29 17-48-28 Juan @ 25:38)
41. O jogo do juan ve a minha equipa a usar uma carta de terra mas nao ve o efeito (torre nao leva dano)
42. O joao ataca a torre do juan com T7 1 vez Video 2016-11-29 17-48-29 Joao@ 36:31 (Video 2016-11-29 17-48-28 Juan @ 36:30)
43. O jogo do juan ve a minha torre a ser atacada em vez da dele
44. [GUI] Selecionar a torre atraves do painel superior deu erro "Null reference exception" (Video 2016-11-29 17-48-29 Joao@ 33:15) [na altura pensei que era de nao ter começado o turno entao nao apanhei a stack ;(] pode ter sido de outra coisa qualquer nao sei =/ - 29/11/2016
45. Malik4 - para o Yuri x)
46. A escolha 1 nao pode vir como counter (pq senao no gui aparece tbm o icon)
47. A escolha 1 deveria ter 1 target heroi a espera d ser escolhido pa mostrar as habilidades (nao tem targets herois porque é considerado 1 counter..)

48. Não faz spawn com o servidor - diz que tá fazendo e não aparece nenhum boneco, o hexágono fica selecionado e já não dá para fazer absolutamente mais nada, é preciso reiniciar o jogo :S - tem acontecido com o Local game até agora



9.5.3 Playtest 02-12-2016

1. para curar escolheu usar e deu um argument out of range (Video 2016-12-02 14-20-21 Juan @ 09:28)
2. O L9 tava a olhar para a lista de objective trigger targets mas no IngameNetworkHelper tava a ser adicionado na lista de objective targets os triggers linha 648
3. `objectiveTargets.Add(MonoObjective.GetObjectiveByKey((int)jsonTriggerHeroTarget.f));`
4. Passou a ser `objectiveTriggerTargets.Add(MonoObjective.GetObjectiveByKey((int)jsonTriggerHeroTarget.f));` Depois vê Yuri se não fez asneira que isto já é do novo servidor
5. [GUI] Por alguma razão a interface não está a limpar os targets depois de usar o trade da lipp, os inimigos afetados ficam selecionados (vídeo 2016-12-02_15-48-12 Joao @ 10:29) até começarem o seu próximo turno (vídeo 2016-12-02_15-48-12 Joao @ 12:04)
6. A kulla do Juan não podia andar (video 2016-12-02_15-48-12 Juan @ 16:30)
7. Não é possível mover a kulla através do cheat panel (video 2016-12-02_15-48-12 Juan @ 18:23)
8. Não era possível fazer end turn (video 2016-12-02_15-48-12 Juan @ 19:10)
9. Depois de fazer login não é possível fazer login novamente e o servidor tem que ser reiniciado
10. Saiu a carta L9 e o Juan escolheu a opção 1 (fazer dano a 1 herói e a uma torre), ele escolheu um herói e uma torre minha, eu vi o dano a ser feito na torre dele e ele viu o dano a ser feito na minha torre (video 2016-12-02_16-17-42 Joao @ 05:45 / video 2016-12-02_16-17-41 Juan @ 05:44)
11. Em geral quando se usa elementos não sincroniza as torres... o Juan usou a L9 para curar um herói e uma torre, a animação apareceu na torre dele no cliente dele e na minha torre no meu cliente e no cliente do Juan a torre foi curada e no meu a vida das torres continuaram iguais (VIDEO 2016-12-02_16-41-08 Juan @ 13:12) / (VIDEO 2016-12-02_16-41-08 Joao @ 13:34)
12. Foi possível matar a torre 2 do Juan duas vezes no cliente do Joao
13. Matar a torre pela segunda vez deu um erro "Coroutine couldn't be started" (Video 2016-12-02_16-41-08 Joao @ 43:59)

14. Roubei uma carta de elementos do Juan com o L3 depois de ter usado as duas fases de movimento no ghora e roubou o elemento mas deu um NullReferenceException (Video 2016-12-02_16-17-42 Joao @ 14:26)
15. O Juan usou L5 e forçou-me a trocar o Kulla2 depois de eu o ter ativado, depois de torcar a habilidade a Kulla nao perdeu automaticamente o buff (Video 2016-12-02_16-41-08 Joao @ 37:32)
16. [GUI] É possível fazer target a uma torre que esta morta carregando no painel superior ou no hex da torre (Video 2016-12-02_16-41-08 Joao @ 43:45)
17. [GUI] o atalho da habilidade so funciona na ultima habilidade desbloqueada
18. Quando faz reset ao ciclo do Malik1, não está a tirar as particulas

9.5.4 Playtest 06-12-2016

1. Kulla não mostra particulas quando é curada
2. Barra do timer activate element só apareceu para a equipa que tinha um elemento
3. Rever bolinhas dos elementos e targets panel
4. Não está a fechar quando se carrega numa bolinha
5. Não está a fechar quando se carrega com o botão da direita (-->)
6. Relógios no header não estão alinhados
7. Verificar drag animation em todos personagens
8. Quando foi ligada a mesa Bamboo de desenhar deu o seguinte erro:



9. Os idles estão a enfiar os bonecos no chão :D
10. L10 - está a gastar 1 recurso a mais quando as duas equipas têm o mesmo número de objetivos vivos
11. Xml estava incorreto. O custo deve aumentar com o número de habilidades e não com o número de objetivos vivos
12. Não foi removido o target do heroi que morreu



13. Após matar um char, deu um erro na corotina do kill
14. Aconteceu na morte do 7º herói, provavelmente falta limpar qualquer coisa nos hexes (existem 6 hexes, quando foi para ocupar o "7º" deu erro)

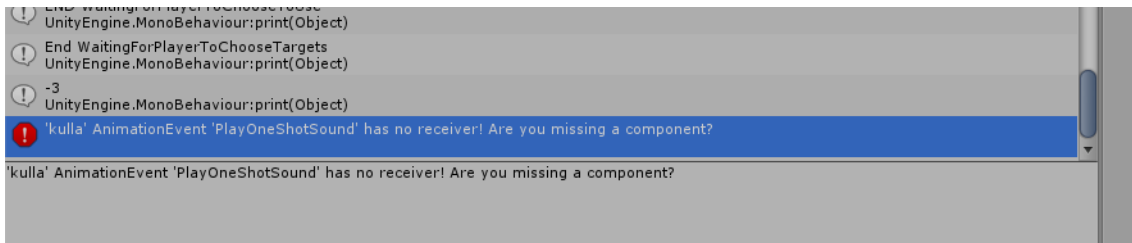


15. QUando força trocar uma habilidade, é preciso limpar a seleção das escolhas



16. É possível fazer start turn enquanto o outro está a usar um elemento
17. Counter da Lipp não pode proteger aliados se for AoE (lobo do ghora)
18. V8 -
19. deve funcionar também na zona das torres mortas?

20. As habilidade de movimento podem usar na torre dos elementos protegidas?
21. A descrição da carta é: Nesta ronda, as peças adversárias não podem permanecer nas zonas das torres vivas, mas podem usá-las para se movimentar. Se alguma se encontrar nesta zona, é movida para a casa mais próxima fora desta pela sua equipa.
22. Logo é só nas torres vivas xD
23. O botão attack tower não atualiza quando se faz start turn
24. As vezes, o botão start turn não atualiza quando muda a equipa que está a jogar
25. As vezes o botão use ability não atualiza quando se faz start turn
26. É preciso desligar todos os ciclos ativos quando uma personagem morre
27. Verificar se o buff da kulla funciona
28. O outro jogador matou-se contra a torre e eu desbloqueiei uma habilidade. Quando fui usar uma habilidade a seguir deu null no updatechoices. A corotina de desbloquear habilidade não desligou por algum motivo
29. Torre morta - remover poder fazer target
30. Erro na kulla quando ela leva dano de qualquer habilidade



31. Mudar de herói enquanto está a ganhar um elemento faz com que a janela dos targets não apareça (elementos instantâneos)
32. Não consigo replicar
33. A1-3 / F1-3 - verificar o que se passa quando não tem targets suficientes
34. [Sugestão] Aumentar o tempo para escolher um poder de elemento da zona dos elementos
35. Os hexagons dos movimentos disponíveis não atualiza quando o target é movido por habilidades/elemento



36. V9 - não pode ser usado se o herói que estiver a jogar já tiver usado a primeira fase de movimento
37. Start Turn não fica laranja e interactable depois do adversário fazer rez
38. O primeiro jogador a jogar numa nova ronda nem sempre consegue atacar
39. Não consegui replicar
40. Malik4 - counter não funciona
41. Falta atualizar a gui do ghora depois do proc do ciclo para mostrar que o ciclo está desativado
42. Matou - desbloqueou uma habilidade mas não escolheu - matou novamente - só conseguiu escolher uma habilidade
43. Não consegui replicar
44. Não existem extra resses suficientes para pôr a equipa toda caso as outras zonas estejam ocupadas
45. Só deve ser permitido fazer rez na zona extra se todas as outras estiverem ocupadas
46. Quando faz spawn, escolhe o primeiro hex, depois muda de personagem e escolhe o hex, ele faz spawn no primeiro
47. Na parte de desbloquear habilidades as k tao 'bloqueadas por polaridade' nao mostram tooltip
48. [SUGESTAO] os numeros serem 'grandes' de 1 passa para 100
49. [SUGESTÃO] Quando sao usados poderes de elemento, o outro jogador tem de saber que foi usado, talvez aparecer um pequeno painel a mostrar numero do elemento, e targets afetados :)
50. [Sugestão] Primeira ação começa o turno
51. Quando anda
52. Quando tenta usar abilidade

53. Quando carrega com o botao direito ou esc nao pode limpar os elementos instantaneos
54. Se o V8 tiver ligado nao se consegue seleccionar os hexes para usar habilidades
55. Adicionar duplo clique na hero bar
56. Lipp 5 - não deixa usar se não tiver targets secundários disponíveis
57. Não consegui replicar

9.5.5 Playtest 24/25-12-2016

1. A habilidade do malik de por a habilidade em cooldown permite por tbm habilidades de utilidade
2. A carta de raios de inverter uma habilidade ta fazendo cosias estranhas, acho que me roubou um elemento
3. Não consegui replicar
4. Ganhar carta instantanea e desbloquear habilidade ao mesmo tempo escondeu o ttargets panel mas a carta de fogo ficou por usar, entao o jogador ja n conseguia jogar mais
5. A lipp do Joao fez uma combinaçao positiva e matou-se contra a torre. O rodrigo tinha feita counter e fez tambem uma combinaçao positiva. No fim do turno, O joao ganhou vento, depois o rodrigo ganhou fogo e ao mesmo tempo desbloqueou uma habilidade o que escondeu o painel dos elementos.
6. Tentei a mesma coisa e deste lado funcionava.. Criei no update um inpput da tecla F3 so pa o pessoal poder usar o targets panel caso de erro enquanto tamos a testar agora, pk senao temd fechar o jogo -> ta no Ingameguicontroller, no Update
7. V1 quando tira da zona das torres nao ve se os hexagonos tao ocupados
8. Pos o Ghora no mesmo hex que a Lipp
9. Ao usar a Lipp3 com o ghora como target nao se conseguia por a lipp como secondary target
10. Tambem n dava para escolhe-la como target principal



11. Não consigo replicar

12. Todos os heróis já tinham todas as habilidades desbloqueadas. Quando a lipp tava a jogar ganhou uma habilidade, o Rodrigo escolheu trocar uma habilidade na kulla e depois ficou preso nela, não voltou para a lipp que estava jogando
13. No hit ou no fly, sorry não tenho a certeza... foi quando testei o V1

```

-2
UnityEngine.MonoBehaviour:print(Object)
! additional objective damage: 0
UnityEngine.Debug:Log(Object)
-2
UnityEngine.MonoBehaviour:print(Object)
! 'kulla' AnimationEvent 'PlayOneShotSound' has no receiver! Are you missing a component?
! 'kulla' AnimationEvent 'PlayOneShotSound' has no receiver! Are you missing a component?
! End Showing zoneelemBarcounter
UnityEngine.MonoBehaviour:print(Object)

```

14. Kulla 3
15. Tem de ser possível usar opção 1 mesmo com traps ativas
16. Verificar movimento no networking quando pisa em cima de uma trap
17. M4 - tem de poder fazer cast com Malik1 ativo
18. Ver todas habilidades de utilidade do malik (cast com Malik1 ativo)
19. T10 ficou ativa sozinha do nada
20. Não consigo replicar
21. T4 -> tem de passar a ser um buff que aumenta o dano em vez de só fazer o dano
22. Rever Kulla4 com V4
23. A4
24. Curou quantidade diferente no jogador remoto
25. Elementpowerguicontroller linha 377. Null quando ganhou L6
26. Não consigo replicar
27. Rever reflect do ghora5
28. Immortal do ghora falhou depois do reflect ter falhado
29. Habilidade atacante foi o ghora3
30. T2
31. Tem de passar a ser um buff que aumenta o dano em vez de fazer dano direto na torre
32. KullaU1
33. Não está a deixar usar gastanto recursos extra
34. O double click para centrar a camera deve desligar quando se carrega com o botão do meio