Mazekeeper
A Browser Real Time Strategy Game

Miguel Francisco Alegra Rebocho de Oliveira

Instituto Superior Técnico

Abstract. This document presents an analysis of the development of a game concept, as well as the sistem developed to support that concept. It is made a contextualization that allow us to understand what are the characteristics, platforms, technology and target players of the developed game. It also explains tests made and implementations done.

Keywords: Virtual Game, RTS Game, Tower Defense Game, Game Concept, Game System, Game Development.

1 Introduction

Humanity has used games to entertain itself since the early ages. Nowadays, with the appearance of personal computers, games are also being created to be played in such technologies. Games in a variety of genres fulfill different ways of applying game concepts to machines. As so, the web is also an environment where a variety of game concepts are also applied. The social interaction created by multi-player online games is becoming as important as the game itself to players [1].

The development of a browser game that incorporate a multi-player component with social interaction was the driving factor for the development of this project. Games genres like RTS (Real Time Strategy) are widely available in browser, because the game concept is easily applied to such technology. Mazekeeper pretends to be a unique RTS tower defense game, importing new features and mechanics to the browser games environment. It presents classic characteristics of a browser game while it contains aspects of RTS and Tower Defense games published for other platforms.

In order to create a virtual game that fulfil the players necessities it is important to analyse the different aspects involved in developing a computer game. At first we should take a look to games that were previously launched. The history of RTS and Tower Defense games might give some inspiration to future game concepts. Games that target the same platform are also of the essence, once they show what are the current game trends. Secondly, the analysis of the psychology of players might also give us important data to deal with. By understanding what kind of stimulus are appreciated by each type of player it is easier to develop a game concept that has a good entertainment level.

In this paper we will start by analysing some related work. After, it will be explained the game concept followed by its system architecture. At last, it will be explored the different prototypes iterations and the user tests results.

2 Related Work

Since the last century, this game genre started to be defined as it is today, but some games gave important contributions for this definition. In 1982 it was released what can be mentioned as the first RTS game. Cytron Masters developed by Ozark Softscape introduced the concept of management of a fleet which more recently turned into managing civilizations. Technosoft launched the Herzog Zwei in 1989 that brought to life concepts like zone controlling and conquering. Despite those game concept innovations, it was only in 1992 that the RTS game genre as we know appeared. Dune II developed by Westwood Studios define the game genre still explored today. This game presented unique characteristics of managing a space colony. The existence of resources allied with the capability of exploring the map, exploiting resources, expanding the territory and exterminating enemies forged what it is perceived today as a RTS game. After dune games like Warcraft and Age of Empires came to refine more this game concept, bringing new mechanics and new features that players could explore.

Tower Defense games are considered a sub genre of RTS games. However this game concept is older than the RTS concept. The first game considered a Tower Defense was Missile Command developed by Atari. Missile command objective was to destroy incoming missiles that would destroy the player city. Other games like Pedro developed by Imagine Software's launched in 1984 explored the concept of making the player defend a crop, by killing the animals that were moving to eat it. Rampart was also a game that reinforced his genre. This game launched in 1990 presented different player phases. Tower Defense games today still present this feature. In Rampart players have three phases to manage. The first phase corresponds to the construction. Players during this phase have to position walls to increase as much as possible the castle territory. The second phase corresponds to the positioning of the cannons which are responsible for shooting the incoming enemies in the third phase. Despite of all games launched before Warcraft III: Reign of Chaos it was this RTS game that turned Tower Defenses viral. The costume game creation mechanism presented in Warcraft III made possible the appearance of many games and even game genres. Moba games were also a costume gameplay mode of Warcraft III and they originated actual games like League of Legends and Dota. Tower Defenses were no exception. This game genre started to became one of the most played in Warcraft mods and subsequently web browser Tower Defense games started to appear.

When developing a game, it is important to know the psychology of players in order to understand what might give them the haste to play a certain game. Games are developed for players, as so, understanding their behaviour and their psychology becomes an important matter. First we can characterise players by

two large groups, hardcore players and casual players. Those groups are differentiated by their attitudes when facing the play activity. Hardcore players are people that plan their life never forgetting the time to play. In other words Hardcore players plan their life in function of playing and of gaming time. Casual players, on the other hand, are users that only play when they face dead times or when they do not have anything better to do [2]. Players can also be categorized by the feelings that urge the play activity. As so, some categories are created by the Brainhex model presented in [3]. From here it can identify 3 types of players that would feet in the developed game concept.

Conquerors is the type of player that seek to fight against strong enemies. This feeling is created by the natural human reactions. Those reactions makes their body produce epinephrine (adrenalin) and norepinephrine. making the last producing excitement sensation which are associated with competitive tendencies.

Master minders seek their pleasure in solving puzzles. Their pursuit can be explained by the thrill involved in finishing to solve puzzles by making the most efficient decisions. The reward feeling of solving such puzzles makes this type of players keep desiring more puzzles to solve and thus improve their skill.

Achievers archetype likes to overcome challenges in order to complete achievements. This feeling is created by the release of dopamine, hormone segregated by the brain in order to induce pleasure. This archetype also likes to be rewarded by their successes and for that Mazekeeper will reward players accordingly to their achievements in each round, making possible the usage of those rewards in the next round.

3 Game Concept

Mazekeeper pretends to be a unique RTS game in the browser game environment. It contains characteristics of a regular RTS, as well as characteristics of a commun Tower Defense. In Mazekeeper, players have to manage their cities while preparing them for an eventual attack. To show a big picture of the game concept, it is presented in Figure 1 the Mazekeeper game loop.

There are three core game scenes, City, Lane and Map. Those three parts of the game represent the main three interfaces that the player will be faced with. Starting by analysing the city component, as in the majority of RTS games, the city interface presents the player with city management options. From here players can improve resource gatherer buildings, recruitment building and progression buildings. Those types of buildings corresponds to the building function. If a building is responsible for providing certain type of resources to the player, then it is characterized as a resource gatherer building, the same logic is applied for the rest of types referred. The city scene also allows players to recruit units that can be used in the lane scene to attack other players or NPCs (non player characters).

Each player is also provided with a lane scene. From here, players position their defenses to protect themselves against other player attacks. The lane scene

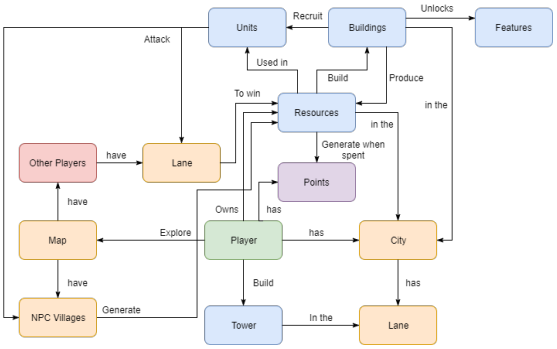

Fig. 1: Mazekeeper Game loop.

contains a matrix of slots. Each slots can hold a defensive building such as a tower or a wall. Those defensive buildings have different properties and thus interact differently to incoming attacking units. By providing such building flexibility, players that are making their defense have to think in the best defensive strategy accordingly to the environment provided. Also, attacking players have three paths that can choose from, since every lane scene contains three entrance points, which increases the strategy depth for attacking players.

The Map scene will give players a world overview. From here players can see the map distribution and choose possible attacking targets. The map scene also shows the NPCs, which are displayed next to player's city. Players from this interface can see other players progression by looking at each player's points. The points system was developed to increase the game competition as well as enabling players to check on others players progression.

NPC villages present a different game mechanic from other browser RTS games. In games like Trbal Wars, NPC villages act like player's cities. In these games, when NPC villages are attacked, players loot a small quantity of the villages resources. Mazekeeper provides a different mechanism. In order to simplify, NPCs does not contain any specific information about the city buildings. Every NPC present a set of 3 attributes, the level of gold prospecting, the level of materials gathering and its defensive level. The gold level indicates the quantity of gold generated by hour. The materials level follow the same principle. However, the defensive level indicates how hard is to conquer that city. There are four sets of three different pre-defined mazes for each level of NPC defense. Accordingly to its level, the system will load one of the corresponding mazes keeping it the same until the level changes. In order to be able to change the level of an NPC city, players have to conquer them. NPC villages present an influence conquering system. When a player attacks a village, it wins reputation with the target city. If that influence reaches fifty percent then the player that holds that percentage becomes the owner of the city, and the rest of the influence starts to grow passively. When a player becomes owner of a city, he is allowed to pay resources in order to improve one desired level. If a player attacks another player's village it

will decrease the amount of influence the owner has. When it reaches less than fifty the city becomes with no owner and when it reaches zero the defense level is also set to zero.

In Mazekeeper there are also special units. Heroes are units that players can recruit in the shrine building. Heroes provide regular units with buffs during the battle scene. This special units were implemented in order to increase the number of possible decisions to each player as well as a rewarding mechanism to the best players each season. Seasonal event is also another feature of the game. At each different season it is applied a new effect to the battle scene that every player as to take into account when planning their defenses and attacks.

4 Architecture of the Solution

A distributed system requires different project modules. Mazekeeper was designed to present three modules, front-end (web page), back-end (game server), game application. The front-end contains the homepage of the project as well as it holds the game application in a webgl player exported from the development framework Unity. The back-end, or game server, is the component that computes every client request. It is also responsible for the database management which can be deployed on a different machine than the server itself. This last feature allows the server replication and thus improving the system redundancy. Finally, the game application is responsible for providing the user interface and all possible game actions to players. Every action done by users using the game application, that requires information or that posts information on the server create a communication channel using a websocket. The project interaction can be checked by the diagram presented in Figure 2.

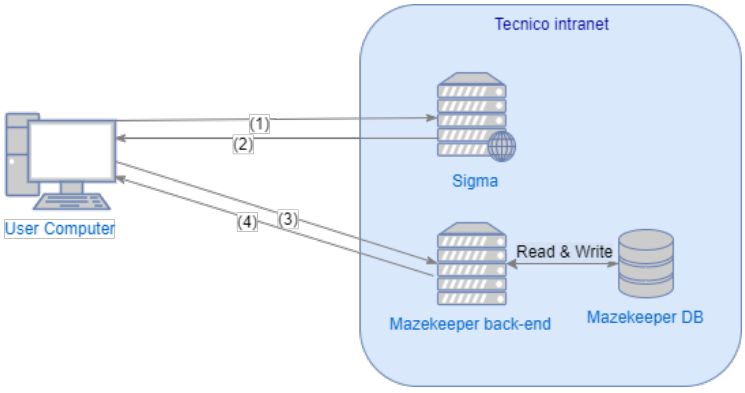

Fig. 2: System Interaction.

The game application was developed using the framework Unity. All scripts developed for the project were written in C#. The game application consists in

a group of four modules. The core module is the Game Manager script. The Game Manager is in charge of all cross cut features as well as registration and authentication processes. Besides that, there are three other modules, each one corresponding to each scene. Taking as example the city scene, it contains the city manager. This script contains all data structures required to set the player state. Along with the city manager it can be found the city displayer. The displayer function is to initialize on the scene all game objects required to reproduce the player's state. The other scenes also present a similar structure, containing the manager component as well as the displayer.

The game server was developed using C#. No frameworks were used to support the development of this part of the project. However some api's had to be integrated in order to provide web communication as well as access to the mongo database. The architecture developed for the game server is shown in figure 3.

From there it can be seen different layers each one having its own access rights and function. The core layer is responsible for every server update. Services as online player resources updates are provided by the core layer of the server. The web listener component awaits for users connections. Upon receiving a new connection, a new thread is generated to fulfill the players request. All web services available will use the data from the cached users and if necessary it will instantiate a new database context to access database information. Like this, every logged user is held in the cache system. This improves the access performance, once the server is not executing any kernel call. The database context component is responsible for every database access, as it was mentioned above. This module uses database objects to translate the database information into runtime objects. Then they are modified accordingly to the requested web service. If those objects were cached before, the old ones are replaced by the new ones in the cache system. If not, the correspondent database object is updated in the corresponding database collection.

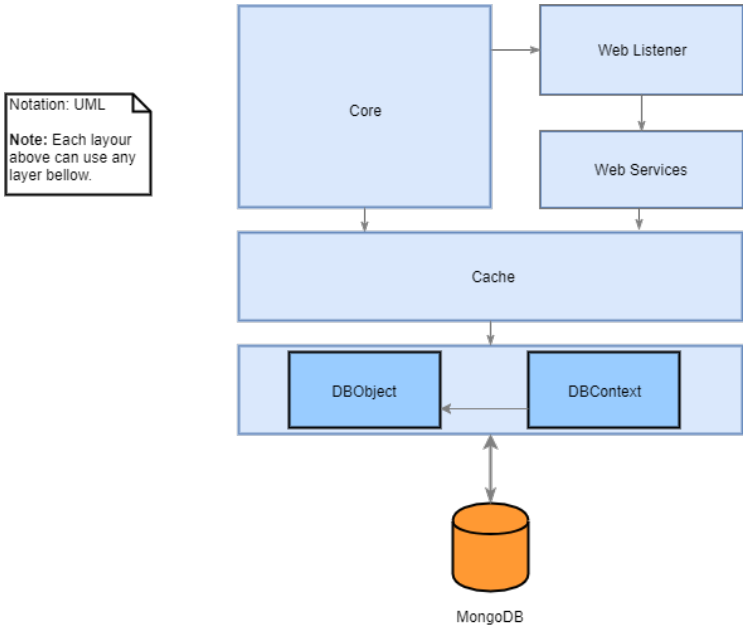

Fig. 3: A server Layered View.

5 Project Development and Testing

5.1 First Prototypes

The project development followed an iterative process. There were two development phases followed by the corresponding testing phases. During the first phase, it was developed three prototypes. Each prototype corresponding to each one of the different scenes. This allowed to test independently all the different game components to be integrated in the final prototype. As the mechanics associated with the three different scenes were difficult to be explored in a low fidelity prototypes, three small virtual games were developed. Each one of them explored a different game scene, from the city to the map. In order to assess the playability of those prototypes two user groups were organized. The first group, the focus group, consisted on a small number of people that besides answering the survey developed to assess the first prototypes also gave their opinion about the game at every development stage. The second group, that includes the first, was a larger test group. From this group is was expected that after playing all prototypes they could provide solutions to some concept problems through the presented survey.

It was expected to retrieve the user's opinion about some aspects of the prototypes presented. As so, the survey was divided into four parts. The first part pretends to characterize the answering user accordingly to its gaming pref-

erences. The second and following groups were developed to give some answers to different aspects of the game, accordingly to each prototype.

From the first prototype it was intended to understand if players would be happy with the flexibility presented when organizing the city defense. This prototype represents the gameplay that players will face when planning their city defense. In here, players are faced with a matrix where they can deploy towers in their most desired positions. There is only one rule that can never be broken. Despite the maze constructed, there must always be a path from every entrance of the maze until the its end. From the survey results it could be concluded that players did enjoy the provided mechanism.

The second prototype tried to explore the inverted game mechanism. In this prototype, players would send the troops to a pre-defined maze and win better rewards the more units could reach the end. The survey presented, also pretended to test some components of this prototype. From this, it was desired the users opinion about the battle deploy mechanism as well as the number of maze entrances. Players shown a clear preference for a deploy system like "Clash Royal" and a number of entrances more than 1.

From the thrid prototype it was intended to understand some features of the management side of the game. This presented a ficticius city that players had to manage, as well as a map view where players could manage their attacks. The survey intended to understand what players thought about the new NPC (Non Player Character) mechanism presented and some usability concerns. Players did have a good acceptance level of the new NPC mechanism provided and so, this was implemented in the final prototype.

5.2 Final Prototype

The development of low fidelity prototypes helped to guide the project towards a final game concept. From the conclusions taken from the prototypes survey a final prototype was planned. It was also important to extract player's behaviour information from the system, in order to be capable of improving it in an iterative way. As so, an analytics system was implemented. With this system, every player interaction is saved in a log, with multiple parameters. Those logs are analysed in order to see possible game concept faults, bugs or game balancing.

The final prototype integrate all three different aspects explored in previous prototypes. From here, it was important to understand how every mechanism would come along with each other. The analytic system allowed us to extract information about all functionality of this prototype. Hence, it will be analysed some information extracted from it.

In order to take some valid conclusions of the information retrieved from the server it is important to make a contextualization of the number of players and number of actions per season.

First it was studied how did the NPC game integration impacted the players' gameplay. To do so, it was analysed the players' activity before and after the update, as well as the number of attacks made.

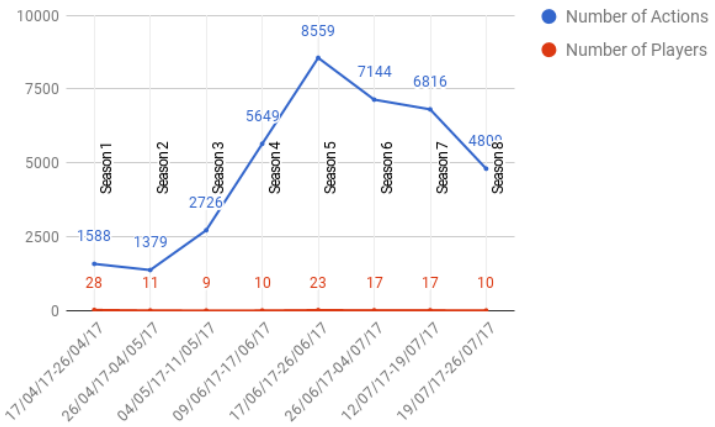

Fig. 4: Game Seasons Analytics.

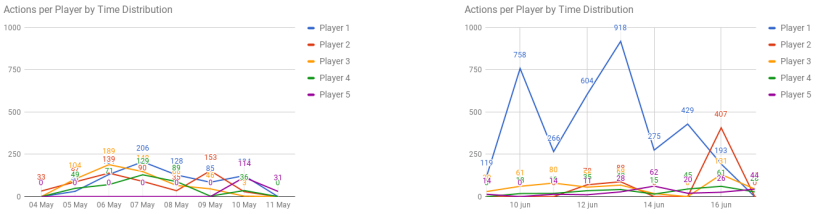

Fig. 5: Comparison between actions per player during the third season, on the left, and the fourth season, on the right.

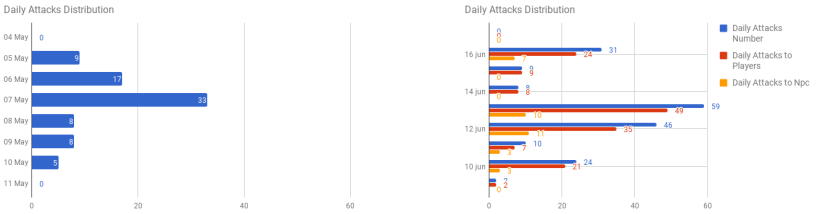

Fig. 6: Comparison between attacks of players during the third season, on the left, and the fourth season, on the right.

From these data it can be seen that there were some problems. First, the lack of new players' protection led to a game unbalance. The fact that one player could attack all the others with no restriction made the game extremely unfair to new players making them to quit when them couldn't make anything to face a stringer opponent. Second, the lack of limitation in the number of villages

players could control also led to a positive feedback, reenforcing the strongest players and prejudicating weaker ones.

To solve those problems some measures were implemented. In order to protect weaker players, it was implemented a system that prohibit players to attack other players that have less than half their points. What regards to NPC villages, it was added a new funtionality to the "Hero Shrine" building, which now increases the radius of the NPC villages players can attack by one radius per level. Finally, units gain a 10% speed debuff for each 1 radius further. This effect stack up 5 times, meaning that a player that attack a village that it 5 slots away from player's main city will have their units moving 50% slower.

The heroes and event update was supposed to bring more strategic depth to the attack. Such update should also influence the player's activity since new content is normally more appealing.

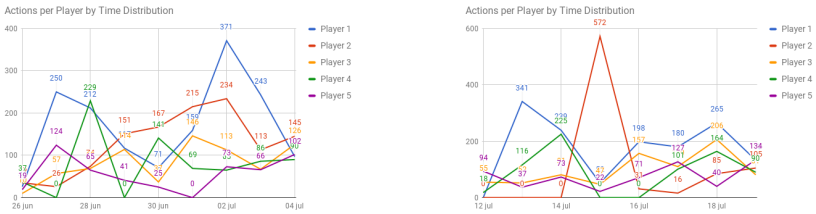

Fig. 7: Comparison between actions of players during the sixth season, on the left, and the seventh season, on the right.

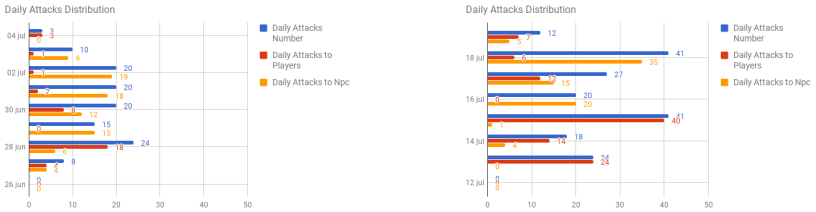

Fig. 8: Comparison between attacks of players during the sixth season, on the left, and the seventh season, on the right.

This suggests that the game became more competitive. Such data proves that the update was a good improvement to the game. Events may also lead to an increase of players activity. By presenting new mechanics to the attack component of the game, we are increasing the strategic depth of the game and so more decision players have to make in order to take the best option available.

5.3 Final Prototype Survey

Even though the system developed allow us to take analytics that help us improve the game, it is necessary to know what is the user's opinion. To extract such information a final survey was prepared and sent to Mazekeeper players. This final survey was designed to infer conclusions about different components of the game, and it was answered by 16 people. One of the focus of the survey was usability. Since no usability tests were developed through the development of this project it is important to assess its usability. As so, the following questions were presented:

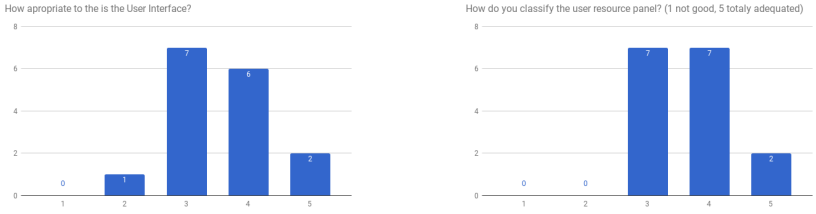

Fig. 9: User's opinion abou the game intergave on the left, and user opinion about resources display interface on the right.

From Figure 10 we can see that the system usability as an average level of acceptance. Users point that the game interface could be better, if though it is not bad. Also some navigation features could also be better.

Other factor that was not possible to assess through game analytics was the consistency of battle simulation. In order to check the player's opinion about the battle consistency the following question was directed to such tematic.

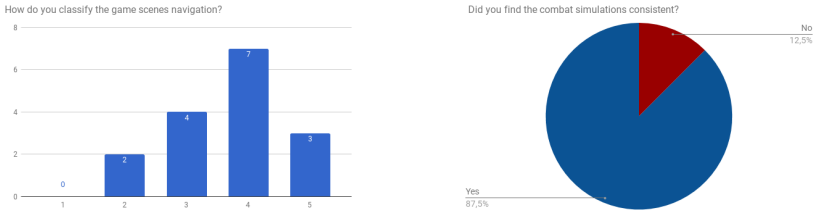

Fig. 10: Right: User's opinion about game navigation. Left: User's opinion about the battle results.

Having 87.5% of the inquired population answering yes, shows that even presenting a battle system that have 2 autonomous simulations a good level of consistency was achieved.

6 Conclusions

The development of a system, such as a web game requires having concerns with multiple aspects. In this chapter we will start by compiling conclusions taken from the development of such system and afterwards present possible future implementations that might improve the game developed.

A web browser game is a much more complex system that might look at a first glimpse. Games have to give players the will to play and to keep playing. As so, interesting mechanics have to be developed, nice art work have to be incorporated and good information must be extracted so that the game can become appealing to as many users as possible. Beyond that, a computer game also must have the best characteristics associated with a software program. From the project presented and delivered, it can be said that all of those things were taken into consideration, as well as analyzed but even though it did not instigate the interest that was thought it would create. One of the reasons might be the low quality of art work used. All interface components have been developed using primary tools or downloaded from opensource communities. With such components players are not graphically attracted to the game and might lead to quiting. Other possible reason is the small community playing. Players nowadays like big communities in the games they play [1], because more competition is available or just because they have more friends playing it.

From this project it was learned that creating a game starts by creating a good game concept. Having a good background work can help to have better ideas. Develop low fidelity prototypes will help to understand if the game concept achieved is fun. It is also important to make as much iterations as possible, so game concept corrections can be made or small details can be polished. Having a group of people that is interested in participate is also a key factor, because the game designer can see if they are motivated or if they have suggestions to the work presented. Players information can be extracted in several ways, and the best way should be appropriated to the solution that the game developer is seeking. Balancing a game is also a very complex process, once some minor alterations can have big impacts on players game play. Developers should have a way of making metrics about different game parameters so they can compare different components and thus understand if those are balanced.

Developing a web game system, was a challenging and fun project. Much have been learned. However, after understanding the complexity of such project it was realised that much more is still to be learned.

References

1. N. Ducheneaut, N. Yee, E. Nickell, R. Moore, ""alone together?" exploring the social dynamics of massively multiplayer online games," *ACM Conference on Human Factors in Computing Systems*, 2006.
2. C. Martinho, P. Santos, R. Prada, *Design e Desenvolvimento de Jogos*. FCA, 2014.
3. L. Nacke , C. Bateman and R. Mandryk, "Brainhex: Preliminary results from a neurobiological gamer typology survey," *Saskatoon, SK, Canada*, 2011.

4. D. Williams , N. Martins , M. Consalvo , J. Ivory , "The virtual census: representations of gender, race and age in video games," *ACM Conference on Human Factors in Computing Systems*, 2009.
5. B. Geryk, "A history of real-time strategy games," April 2011.
6. C. Poladian, "'clash of clans' maker supercell posts $2.3b in revenue, $930m in profit for 2015 as growth slows," March 2016.
7. Goodboy Digital, "Pixijs — the html5 creation engine documentation," 2017.
8. Unity Technologies, "Unity documentation," 2017. Documentation last consulted 10th of August 2017.
9. Epic Games, "Unreal engine documentation," 2017. Documentation last consulted 10th of August 2017.
10. Springer, ed., *Relational Algebra and SQL. In: Introduction to Constraint Databases.* Springer, New York, NY, 2002.
11. J. Kurose, K. Ross, *Computer Networking - A top-Down Approach.* Pearson, 2013.
12. Z. Parker , S. Poe , S. Vrbsky , "Comparing nosql mongodb to an sql db," *Proceedings of the 51st ACM Southeast Conference Article*, 2013.
13. P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford, *Documenting Software Architectures - Views and Beyond.* Addison-Wesley, 2010.
14. M. Azevedo, *Teses, Relatórios e trabalhos escolares.* Universidade Católica Editora, 2004.
15. C. Izurieta , N. Nurseitov, M. Paulson and R. Reynolds, "Comparison of json and xml data interchange formats: A case study," *Department of Computer Science*, 2009.
16. M. Buro, "Real-time strategy games: A new ai research challenge," *Department of Computing Science, University of Alberta, Edmonton*, 2003.
17. P. Avery, J. Togelius, E. Alistar, and R. Pieter van Leeuwen, "Computational intelligence and tower defence games," *Center for Computer Games Research IT University of Copenhagen*, 2011.
18. "Retail principles for free-to-play games," *GameSparks*, 2016.
19. J. James Garrett, "Ajax: A new approach to web applications," *semanticscholar*, 2005.
20. *MongoDB Documentation.* https://docs.mongodb.com/. Documentation last consulted 17th of August 2017.