Designing and Implementing a browser RTS

João Pedro Lopes Ferreira
joao.lopes.ferreira@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

June 2019

Abstract

The video game genre that is recognized as RTS, had many stages of evolution since the beginning of it's creation. As the technology evolved this type of game can nowadays be played in different types of devices. There is still few RTS games that can be played with more than one different device.

This document presents the solution to design and create a RTS Game, that is playable on both personal computers and mobile devices. The game conceived on this project uses HTML5 and JavaScript as programming languages ND is playable in multiplayer mode that multiple players can compete between themselves (1 versus 1). This game also contains the 4X elements present in the traditional RTS games ("eXplore, eXpand, eXploit and eXterminate").

Keywords: RTS, Multiplayer game, Phaser.io, Client-Server networking, Cross-platform.

1. Introduction

This is a war universe. War all the time. That is it's nature. There may be other universes based on all sorts of other principles, but ours seems to be based on war and games. - William Burrows[4]

The term RTS was introduced in BYTE magazine in 1982 [9, pg 3, 124], however it's Brett Speny that is credited with the creation of this term with his game Dune II[8].

Strategy games challenge the player to achieve victory, by planning and using different strategies that are used against one or more opponents[2, pg 419]. This genre distinguishes strategy games from puzzle games that call for planning without the existence of conflict, and from construction and management simulations that doesn't have influence against opponents. Strategy games usually has as a key goal defeating all enemy forces, so the majority of strategy games are military games with some differences between each other. Victory is obtained by superior planning and taking the best possible actions, where the elements of chance or luck are not determinant. Other challenges such as tactical, logistical, economic, and exploration challenges can also be present.

This type of games falls in two main sub-genres: Turn-Based Strategy and Real-Time Strategy. In a Turn-Based Strategy, players can consider what type of actions they want to do in the game, considering the benefits between each available actions and choosing the most optimal ones during it's turn. The player doesn't have to worry about the time that is passing during the consideration of it's actions. Real-Time Strategy Games who were developed after the Turn-based genre, add time to the game for pressuring the players to choose the optimal actions at a fast pace[2, pg 420].

The RTS Video Game genre can be played in both personal computers and mobile platforms. In 2015 ESA released a study about the state of industry and video game industry in the USA[1] during the 2014 year. In this study is reported that 62% of the population plays video games on personal computers, 35% on their smartphones and 31% on wireless devices. Also 37.7% of the computer games played were of the Strategy genre. Moreover 54% of the gamers played at least a multiplayer game once per week.

According to the Independent there is at least one mobile device for each living person on planet[3]. These devices have powerful graphical capabilities that enables to run video games. Those games can be installed and downloaded in this devices by application stores (App Store, Google Play), or be played in a mobile browser.

The main **Objectives** present in this thesis is the creation of a RTS game using HTML5 and JavaScript languages, working in personal computers and mobile devices (cross-platform), containing the traditional 4X elements present in this game genre ("eXplore, eXpand, eXploit and eXterminate") and being able to be playable in multiplayer mode.

To be able to accomplish the main goal of this thesis, **User Tests** were made with a chosen Focus Group made of people with heterogeneous

characteristics. These tests were performed to observe the game experience that is being proportionate for the participants of these type of tests, with the goal to improve that experience and check if the desired objectives purposed at the start of this project were accomplished. For gathering the results made by those tests, there were used three types methods. Those methods were Observation, Questionnaire and Server Text Logs. Those methods would be applied on two sets of test experiments, corresponding to the Alpha and Beta prototype made for this thesis project.

2. Related Work

Regarding the elements that are part of a RTS: The **theme** of a strategy game usually is derived from the primary activity that takes place in the game. While there can be differences among RTS games, most of them fall into different categories of games of conquest, trade, or exploration.

Since the player acts as a military commander of a faction or nation in RTS games, he must deliver orders to it's troops so the player can build an army to defeat it's opponent. Those troops are treated **Units** existing various types of them, with some of them not being specialized in combat but contributing to the growth of the player's army. There are also **Structures** who permit the construction of units, development of new technology or defending against enemy troops. To be able to produce these two entities, the player must use the currency of the RTS games which is **Resources** by harvesting them. The resources are spread across the **Game Map** which reflects the world of the game. At a start of a RTS game, the game map is shrouded by black clouds being called **Fog of War**, limiting the vision of the player and having to move units or construct structures on those areas to become revealed. One aspect that is characteristic of this type of game is the **4X** elements, being mentioned for the first time by Alan Emrich in September 1993[6, pg 92-93]. The meaning for those for elements are: *Explore* means that players send units across a map, to reveal surrounding territories and enemies; *Expand* means that players conquer new territory by creating new structures on it, or extending the influence of existing structures; *Exploit* means players extract and use resources to improve their nation; and *Exterminate* means attacking and eliminating other players, mostly trough military battles.

The majority of RTS game were released to the computer platform, since the mouse and keyboard of it would help the players to deliver orders to it's army, with more speed and precision than the input events made on mobile devices. Also the mobile devices did not had the computational power available for the computer platforms, not being the appropriate game device to play RTS games which are very compute-intensive. Nowadays mobile platforms are severely more developed, being able to use the internet on mobile browsers. With this it is possible to have the game of this thesis being **Cross-platform**, being played on both devices types referred above.

To do so there would be needed to use two programming languages. The first one being **HTML**, or called HyperText Markup Language. HTML allows the creation of websites. HTML documents are made with HTML elements, that have a start tag and end tag that are written with angle brackets and the context between them. With the use of these tags it's possible to manipulate various elements that can be part of a web page, such as a title, a header, a paragraph, an image, a video and many others. This language can also embed scripts written in other languages, for performing functions on web pages that HTML alone can't do. The second programming language used on this project, is one of the languages that can be used by HTML, which is **Javascript**. This language first appeared in May 23 1995 developed by Brendam Eich, and it has been standardized in the ECMAScript language specification[7, pg 2]. Javascript is commonly used with HTML to add client-side behaviour to web pages. With the help of the Domain Object Model it's possible to add interactive events to a web page, linking scripts to a HTML page in the $<$script$>$tag.

For helping to create a RTS multiplayer game using these two languages, a HTML framework that could help to implement a game to be played on multiple platforms was used. The framework is called ***Phaser.io***, or simply called *Phaser*. This framework could solve many issues that are present with the development of cross-platforming games, such as solving the screen resolution, provide multiplayer options, load sprites, sounds and animations and others. The main reasons that *Phaser* was the chosen framework for this project, was the numerous online examples available for it and the active and helpful community present in it, that were greater than any other framework that was researched.

3. Game Design

In this section there is a description on some elements that were part of the Game Design of this project thesis, the game being called ***Planetary Conquest***.

There are two available **Races** to be played on this game: the **Humans** and the **Orghz** who are vile green creatures. The main objective of this game is to destroy every unit and structure that the enemy race disposes. The game window of *Plane-*

tary Conquest is in a top down view with 2,5D perspective or called **isometric projection**. With this type of projection it's possible to represent shapes close to a three-dimensions group. However only a side of a game object is shown per time frame. This type of perspective is referred as 2,5 perspective, since it is an illusion of three dimensions depth.

There are 4 types of **Units** that exist in this game, with each race having one unit for each type. The military units of each race have different combat attributes, in order to differentiate them. The types of units are the following: **Peon** which is weaker in combat capabilities, but it is vital to the growth of the player's army. It is the only unit that can gather resources, construct and repair structures. The **Melee Infantry Units** are the cheapest units that the player can produce, being able to attack units and structures from a short distance. **Ranged Infantry Units** are the other type of infantry present in this game, being slightly weaker than the former military unit presented, but it is capable of attacking from afar. The **Mounted Units** are slightly stronger than the infantry units, but they are mounted by horses, having a greater movement speed. The last type of unit **Flying Units** is capable of flying over the terrain of the game map, ignoring the collision with any unmovable game object. Only units that have ranged attack type or who are flying units as well can attack this type of unit.

Regarding the **Strcutres** there are 8 types, which contribute to the production and strengthening of a military army. Starting with the **Command Center** who is vital for the growth of the player's race, is responsible to the production of peon units and to receive the resources gathered by them. Then there is the **Barracks** which is responsible to the production of infantry units, **Factory** for the production of mounted units and **Flying Nest** for the production of flying units. There is also a defensive structure which is the **Tower** who attacks enemy forces, and the **House** who permits for the player to increase it's maximum population in order to be able to produce more units. For strengthening the player's army there is the **Research Building** which permits to increase the units damage corresponding to their attack type and improve their armor points. Also there is another structure that permits to strength the units selected by a player temporarily, which is the **Power Building**. This structure permits the use of the player's race **Super Power** which is a feature of *Planetary Conquest*. After this structure is produced, by waiting for a specified time it is possible to temporary strength the player's selected units, at a magnitude greater than the other upgrades that the player can choose. The Super power for both races have different applications, for the Orghz is increasing the damage inflicted and for the Humans is reducing the damage received. There is another feature regarding the available upgrades, which is the existence of **Unique Upgrades** for each military unit that can be researched on it's corresponding military structure, and the race and tower upgrade which can be researched on the Command Center.

The **Economy** of this game is composed of three resources: **Crystals**, **Nitrogen Liquid** and **Hydroxygen Gas**. Those resources are spent by the player in order to improve it's military power. Those resources are gathered on **Resource Deposits**, one for each resource that are spread across the map. There is a special set of those deposits which is called **Ancient Resource Deposits**, that doubles the amount of the quantity of resource that is extracted each time by a peon unit, which is 20; compared to the other deposits which is 10. The **Hydroxygen** resource has different uses for each race. On the Orghz it is spent to produce Units and Structures, and to develop new technology. For the Humans it is used for their units breathing, being spent over time during a 6 seconds timer, with the consumption increasing regarding the amount of the population an Human has. If it is not possible to consume **Hydroxygen** for this behaviour, the Human units will take some damage to their health points.

Another feature present on this game is the existence of **Monsters**, that are present in small groups spread across the map. There are 3 types of monsters: The **Ogres** being a Melee Infantry type, the **Archers** being a Ranged Infantry type and **Gnomish Airplane** being a Flying ranged type. Each monster can be interactive with the players units, providing the use of 3 **Monster Options**: The **Trade** option which enables the trading of a specified quantity, of one type of resource a slightly bigger quantity of another resource. The **Recruit** option for receiving 3 copies of the type of monster that was being negotiated as units. The **Pillage** option which enables to try to kill those monsters receiving some resources in return.

Planetary Conquest has the 4X elements present on it. The player can *Expand* by constructing new structures, it can *Exploit* by extracting resource locations, or interacting with monsters and it can *Exterminate* by killing enemy troops or monsters. The player can also *Explore* the game map, to gather resources in other Resource Deposits or to interact with monsters encampments.

The figure bellow represent the interface of this game found on mobile devices with the elements discussed in the Game Design section.

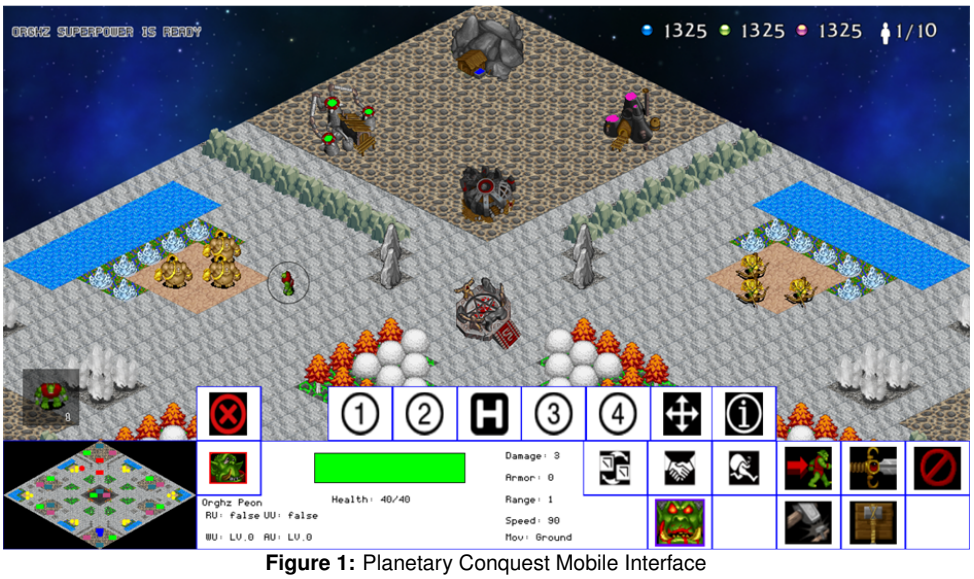

Figure 1: Planetary Conquest Mobile Interface

4. Game Architecture

This section describes the **Game Architecture** that is part of this project. Starting with the **Technical Limitations** of *Planetary Conquest*, a **Game Server** is needed in order for players with their game devices to access it as clients, to be able to play this game. Also it has an important role in maintaining the clients synchronized and being authoritative over events that unfold in the game matches. The server uses the library **Socket.io** to enable real-time, bidirectional and event-based communication between the browser and the server; and **Express** which is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. The game framework **Phaser.io** is needed to run this game on personal computers and mobile devices, using an **Internet Browser**.

The users of *Planetary Conquest* are expected to know how to access to the Game Server domain using their game device for it and they must be able to input commands to the game in order to do the game actions they desire. Also they want the game inputs to have a fast response time, with the use of an interface that is easy to navigate with.

As stated before a Game Server is needed to connect the players to *Planetary Conquest*. The server will have all the information about the game match, simulate some events belonging to it and have complete authority about what it is enfolded between the players actions. In short the server is in charge of providing data and services to one or more clients. In the context of game development the most common scenario is when two or more clients connect to the same server, the server will keep track of the game match as well as the distributed players.

The type of connection for the communication between the Server and the Clients is TCP (Transmission Control Protocol). When data is sent through this protocol, the application running in the source machine first establishes a connection with the destination machine. Once a connection has been established, data is transmitted in packets in such a way that the receiving application can put the data back together in the appropriate order[11]. TCP also provides built-in error checking mechanisms so that, if a packet is lost the target application can notify the sender application, and any missing packets are sent again until the entire message is received. There are still disadvantages with the use of this protocol, namely the time elapsed of the replies sent by the server to the connected clients is bigger, comparing with other commonly protocol used named UDP (User Datagram Protocol). Using TCP is still preferable as the communication protocol between the game Server and Clients of this project, since this protocol guarantees that every message is sent to the clients in it's entirety.

To guarantee that the clients are synchronized with the server, and the messages sent by the server are received to all the clients at the same time without major latency problems, it was used an architecture known as ***lock-step* networking model**[10, pg 382]. The server will achieve this behavior by running a game timer at 10 clock ticks per second (100 ms). The player's client also runs a game timer, at the same pace as the server. Each time the client timer ends he will send to the server his own game tick, being updated to the client's corresponding socket. When a command request is sent by a client, it will be stored in a

command array that the server has. Each time the server game timer ends he will send the response of the requested commands to the clients. Those commands have specified the current game tick on them. The clients will execute the commands response, corresponding to the game tick received. Since the server needs to execute the response to the commands for all the players at the same time, it will need to wait for the commands sent from all the players to arrive before stepping ahead to the next game tick, which is why it's called *lock-step*. If there is a least one client whose game tick is lower than the server's game tick, which could be caused by latency issues, the commands stored by the server will not be send to the clients. The server has to wait for another timer cycle, to check if the clients who had their game tick lower than the server's game tick are now synchronized with the server's game tick, in order to the response of their commands to all the connected clients.

The *Lock-step* method demonstrated how it was possible to have the clients synchronized with the server, and the server messages be relayed to all the clients at the same time. However there is still the emergence to guarantee that no types of cheating happens. To do so the server needs to be **authoritative** over the events of the game match played by the clients. With the server simulating the game match state, the clients do not have to worry about being cheated or receiving incorrect messages about the events of the game. One example of a server using it's authority is in unit vs unit combat. The order for these units to combat is made by the clients input and the animation of those attacks as well. However the calculations of damage on that combat are simulated by the server only. The server sends information to the client about the amount of health an unit has lost by enemy attacks, and when it has to die. In summary the game state of a game match is managed by the server alone. Clients send their actions to the server. The server updates the game state periodically and then sends the new game state back to clients who just render it on the screen.

After completing the discussion for the architecture that is part of the Game Server used in this project, we will discuss the architecture that is part of the game framework used on this project *Phaser.io*. Game states are files that have different parts of the *Planetary Conquest* game included on them. When one part of a state functionality is completed, it requests the start of a following one. The first state of this project is *game.js*, responsible for creating a *Phaser* game instance with a size and render mode specified. Following this state is *boot.js*, which is responsible for the initialization of the game screen, depending on the game device used and other types of initialization such as the type of physics to be used or enabling input events. The next state is *load.js* responsible for loading to the game various assets used such as images, spritesheets, text fonts or JSON files containing information about the game map. The final game state is *play.js*, which is the game manager of *Planetary Conquest* and has various functions such as initializing the interface, respond to client inputs or receive and treat the information sent by the server. The three interactive game objects (Units, Structures and Monsters) also have a class that contains their functionality, being *unit.js*, *building.js* and *monster.js*.

5. Implementation

This section describes what were the mos important tasks that were implemented in this project, in order to fulfill the main objectives purposed for *Planetary Conquest*.

The first task implemented was the use of **Spritesheets**, which are images that contain several other small images that can be part of an animation of a game object. Combining the small images in one big image improves the game performance, reduces the memory usage and speeds up the startup time of the game. The spritesheets used for this project can be found on the website **Sprites Resources**, and are part of the RTS game *Warcraft II: Tides of Darkness*. Those spritesheets were not appropriate to be used on *Phaser*, since each image did not contain the same dimensions and were not evenly separated. The first step used to get an appropriate spritesheet to be used was using the program **ShoeBox** to extract each small image that was part of that spritesheet. Then those images would be loaded as a pile on **GIMP**, being modified to have the same width and height and being placed on a new spritesheet. Now *Phaser* could load the modified spritesheet, by knowing the size of each frame that would be used on a game object animation.

The second task implemented was the **Game Map**. It was used the tool **Tiled** which is a 2D map editor both for orthogonal and isometric perspective, to create a game map image. The map is divided in a grid by tiles, containing 45 rows and columns of them. Each tile measure is 64x32 pixels, which has appropriate dimensions to be used on an isometric game. The game map would be loaded on the game load state - *load.js* as an image, then it would be placed in the game manager state - play.js. Each tile position would be referenced to a matrix being also 45x45, where there could be one of two possible numbers: 0 specifying a non-occupied position, and 1 specifying an occupied position.

There was also the use on an **isometric plugin** available for the *Phaser* framework, to use a set of operations regarding isometric projection. With the use of this plugin, it was possible to calculate the position or velocity of a moving game object on a isometric game world.

For units and monsters being able to have intelligent movement, walking from a start to an end position using the shortest distance possible, a pathfinding algorithm was used called **A-Star movement**. This algorithm calculated the shortest distance between a starting node and an end node of a node graph. I used the following equation:

$$f(n) = g(n) + h(n) \quad (1)$$

The objective of this function is to have the lowest **f** value possible, which will indicate the shortest path possible for one node to another. The parameter **g** means the movement cost to move from the starting node to a given square on the grid, following the path generated to get there; and the parameter **h** is the estimated movement cost to move from that given square on the grid to the final destination, which is referred as an heuristic. With the use of an heuristic it would be faster to calculate the shortest path possible. The heuristic used for this type of movement that permitted diagonal movement between nodes was the *Euclidean* heuristic, that is represented by the following formula:

$$h(n) = \sqrt{(goal.x–n.x)2 + (goal.y–n.y)2} \quad (2)$$

At some point of the project's development it was needed a data structure that would contain the position of the game interactive objects. It would be helpful to search for nearby game objects that were close of a specified game object, instead of comparing the distance between every existing game object present on the map. The method used for this data structure is called **Spacial Hashing**, that is a process by which a 3D or 2D domain space is projected into a 1D hash table. For this project it was used *arrays* instead. There were two arrays used for this method, one for units and monsters, and one for structures. Each array had 64 positions being referenced as a matrix of 8 lines and columns respectively. Each position of the array or called by this method *bucket* represented a portion of the map, in this case a 1 by 64 part of it. On these buckets it is kept the interactive object, whose sprite bounds intersect with the regions they are specifying. With the interactive game objects position being referenced and updated on this method, by checking the adjacent positions of the buckets that a game objects was part of it, it could be easily found the other game objects who were close to it.

Since this game has cross-platform functionality, using mobile devices as one of the compatible game devices for playing *Planetary Conquest*, the **Device orientation** of this type of device needed to be fixed on landscape position. Also since there are various screen sizes for the mobile platforms, the game size was scaled according to the screen measures of them.

The game server also employed other types of **authority** not discussed yet, such as the validation to produce units and construct structures, research of new technology, simulate the movement of units and monsters; and the behaviour of the *Hydroxygen* resource for the Human race.

There were some measures adopted to increase the **game's performance** such as the use of prefabs for Units and Monsters, the rendering option used being Canvas instead of WebGL, use of Bitmap text instead of normal text and the use animations when game object are defeated instead of particle systems.

Some behaviours common to the RTS genre such as the **Fog of War** or **Collision and Avoidance** of game objects, could not be implemented either by prejudicing the game's performance or not being able to be simulated by the Game server calculations.

6. User Tests

This section presents the **User Tests** performed with the Focus Group of this project, and the results that were obtained with an analysis of them. These tests were performed to observe the game experience that is generated by the users playing the game's project. They also helped to identify the components that are contributing more to the game experience, and to find an equilibrium on parameters that were be analyzed[5, pg 253].

There were 20 people that were part of the **Focus group** gathered, having participated in 13 user tests. Each person had different characteristics between each other such as age, job, education and gaming experience with RTS and computer and mobile games in general. It was important to have a group of people with heterogeneity characteristics, in order to have more different gaming experiences within the tests performed. There were two sets of tests performed, corresponding to the **Alpha Prototype** and **Beta Prototype** of *Planetary Conquest*. For the first prototype, the User tests were mostly performed in the developers house, since the participants were already accustomed with it not having any problems or anxiety to that place. For the **Beta Prototype** most of the tests were performed on the **Instituto Superior Técnico - Campus Taguspark** during the MOJO 2019, where *Planetary Conquest* was one of the games showcased during this event.

Each test was performed by two participants, one being the Orghz player and the other being the Human player. Before the players started the test it was explained why these test were being performed and what they would do in the test: The participants would be playing for the maximum of 15 to 20 minutes (Alpha and Beta prototype respectively) for trying to defeat their opponent. There were also 5 minutes used before the tests that were used to explain the contents that could found in game, and explaining some basic actions that the players could do such as gathering resources with peon units, constructing a structure, choosing monster options and attacking an enemy unit.

For analyzing each set of tests 3 methods were used: **Observation**, **Questionnaires** and **Server Log Files**. The first one **Observation** consisted on looking what the players were doing on each game device screen, and taking notes about some events during the experiment; such as their visible emotions, if a game event was not behaving properly, if there were problems with the usability of the interface and others. The second one **Questionnaires** are set of questions that would be answered by the participants at the end of the User tests performed. The questionnaire contains 4 sections: The first section has questions about the characteristics of the tester such as age, job, studies and the experience of playing RTS and computer and mobile games. The second section contains closed questions about the game elements of this project such as the race played, the opinion about the usage of each resource available, the monster options, resource costs of units and structures. The third section is related to the game experience, having questions on how focused the player was on the actions he was performing, how quickly the player could thought about the actions he wished to perform, his opinion about the game interface and controls, the feelings he had when it was playing the game and if the game posed some challenge to his gaming capabilities. The last method **Server Log Files**, consisted on the recording of some events that were part of a the game match between 2 user testers written in text files. This method was used to record some game events such as unit and structure production or combat results.

The **Alpha Prototype** of this game was the first prototype to be tested. At the time this prototype only enabled the users to construct the first four structures (Command Center, Barracks, House and Tower) of *Planetary Conquest*, the research of upgrades was not available for any unit or race, Ancient Resource Deposits were not placed at the middle of the map, Human Hydroxygen timer was not functional and the monster Gnomish Airplane was not present on the map. Also since the game design was at early stages of development, each type of units and structures available had identical resource costs and attribute values regardless the race chosen. The reason for keeping this project with few features available was for observing what would be the most important actions that the users would do, with limited actions and time available.

By applying the Observation method on the Alpha Prototype tests it was noticed that the Users were a little anxious at the start of the experiments, with the major cause being the 15 minute time constrain for the duration of the tests for defeating their opponent. However they began to be more relaxed after seeing their military power increased by the actions they were performing during the game match. There were some help required by the participants, in some aspects of the game that were not understood completely during the pre-testing phase, but there were few doubts asked by them being also quickly solved. Other factors observed included that the players were not much attentive to what their opponent would be doing, being more concentrated on their own actions; and some game control options such as the button used for selecting idle peons, and using keyboard shortcuts were rarely used.

Regarding the Questionnaires applied to these tests, in the first section it could be seen that the users differed on the experience in playing RTS games, but most of them were at least experienced in playing computer or mobile games. About the second section, referent to the actions performed in-game most users found that *Crystal* were much more important to use than the other two resources, the map size was not adequate to construct numerous structures; and from the 3 available monster options the *Recruit* option was quite used and useful to the users, while the other two not so much. For the third section it was possible to observe that most players were focused on the actions they were performed during these experiments, although some of them were distracted during their actions. Most of the players knew the actions they wanted to perform and knew how to operate with the game's interface for the execution of those actions. Regarding their feelings the users liked to play the game and felt that it posed some challenged to their game capabilities. On the fourth and final section the players thought that the game needed to include Fog of War behaviour and the interaction with the monsters could be differently made. Some of the elements analyzed using the server text logs for this experiment are represented as boxplot graphs on the figure bellow.

The conclusions obtained by analyzing these files were the following: Regarding the **Units pro-**

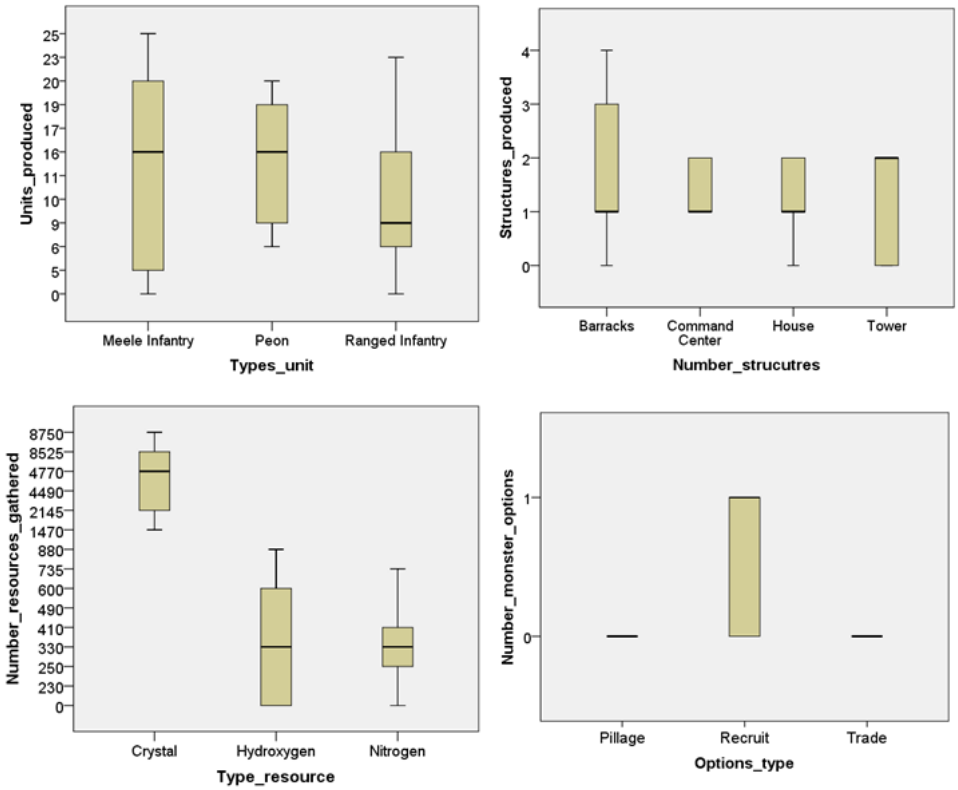

Figure 2: Alpha Prototype elements analyzed

duced the average of peons and Melee infantry units were close from each other being 13.5 and 13 respectively. The number of average ranged units was 6.8 being lower than the other 2 units, but was still substantially used. Regarding the **Structures produced**, the average number of Command Centers and Barracks were the same being 1.1, and the houses was the structure most produced with 1.6 since the Tower average was equivalent to 0.83. For the **Monster options** only the **Recruit** option was chosen 4 times for the Ogres and 1 times for the Archer. Looking at the **Resources Gathered**, Crystals were by far the most resource obtained with an average of 2970.8, with the Nitrogen and Hydroxygen falling much shorter being 195.8 and 33.3 respectively. Finally the average duration of each match was 12 minutes. The figures above show some boxplot graphs made by the results of those elements.

There were some conclusions obtained with the results of these tests. Most of the expected results were met, the players could effectuate most of the actions they desired with easiness for trying to defeat their opponent, and they weren't many difficulties presented to apply those actions. However there were some issues that could be revised observing the methods used for these tests. The ones that suffered a change were the monster options, since the only option chosen was **Recruit**. That option had it's resource cost increased during the design process of these options. The other complain was about the map space available. Instead of changing the map dimensions the function responsible for placing structures on the map was changed. At the time a 2x2 structure size would check 4x4 positions on the map, for being able to be certain that the adjacent tile positions of that structure would never intersect with occupied positions. This approach was abandoned and now each time a 2x2 structure would be produced, it would check the corresponding 2x2 tile positions that were tested to be placed upon.

The second set of tests were performed on the **Beta prototype** of **Planetary Conquest**. All the elements discussed in the **Game Design** section such as Units, Structures, Upgrades and Monsters were implemented on this prototype. The usage of the Super Power, the placement of Ancient Resource Deposits and the existence of the Hydroxygen Timer for the Human race were also functional. The results expected in these tests would be the production of some of the new units and structures added, the use of upgrades, Super Power and monster options for the Gnomish Airplane. The conditions were the same as the previous tests made in the **Alpha prototype**, however the tests had a time duration of 20 minutes instead of 15. There were tests performed in all the proposed platforms: computer devices and mobile platforms (Android Smartphone and Tablets). Most of the

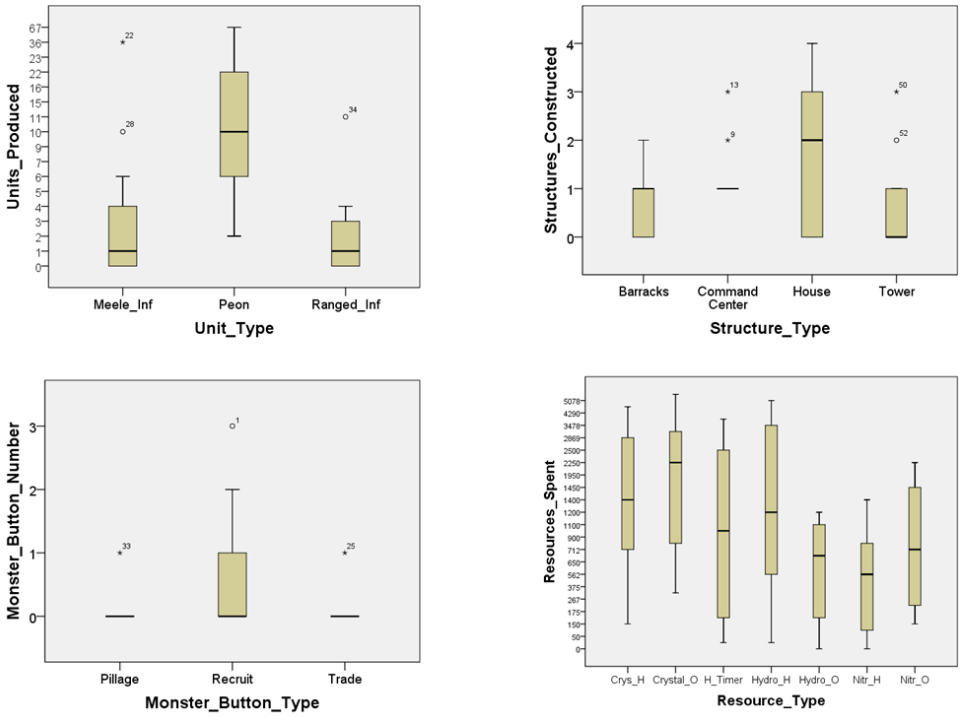

Figure 3: Beta Prototype elements analyzed

tests performed were on Android Tablet devices. There were 15 participants for this set of tests, 2 which participated in the Alpha prototype and 13 who participated during the MOJO 2019 event.

Since the majority of the tests was performed on mobile platforms, it was important to observe how the users could operate on the mobile interface of *Planetary Conquest*. While the users had some doubts at the start of the experiments, they were able to quickly grasp the functionality of mobile controls. Also most of the users understood how to make the basic actions of this game quickly, and was able to form a military army for trying to defeat their opponent's army. Most of the features present in this prototype were used, even if sporadically.

Regarding the questionnaire method used on this experiment, it could be observer on the second section that the use of **Nitrogen** and **Hydroxygen** resources, had more importance on the development of the players army in this tests than in the previous one. Also players suggested that the mobile game's interface could be changed and the map size could still be bigger.

By observing the data including on the server text logs, the conclusions were the following: Regarding the **Units produced** per game, the average of peons produce was of 16.625, being by a large margin the most produced unit during these experiments. The average of Melee and Ranged Infantry was of 4 and 1.87 respectively, much lower than what was obtained on the Alpha prototype tests. Regarding the **Structures produced** the average of Command Centers produced was of 1.1 and the Barracks structure of 0.68, again shorter on what was obtained during the Alpha tests. The House structure was the one being more constructed with an average number of 1.75. The Tower structure had an average of 0.68. Referring the **Monster options** the **Recruit** options was chosen 8 times, twice the amount of the previous tests with Ogre and Archer monsters being both recruited by 3 times, and Gnomish Airplane two times. The **Trade** and **Pillage** options were only chosen once. Observing the **Resources Spent** for each race, **Crystals** was still the most used resource with an average quantity of 1229 and 1187.5 for the Orghz and Human race respectively per game. The **Nitrogen** had an average of 480 and 268, and the **Hydroxygen** resource was spent much less on the Orghz race with an average number of 318.75, than on the Human race with 924.75 respectively. Most of the **Hydroxygen** spent for the Human race was on the **Hydroxygen Timer** behaviour, with some of it being used on the Monster options. Above there are presented boxplot graphs regarding the results of those elements, on the Beta prototype tests.

The results obtained for these tests mostly corresponded to what it was expected. *Planetary Conquest* could be played on both Personal Computers and Mobile Devices. Also the users did not presented severe difficulties on understanding the mobile interface of this game, and knew how to

navigate with it in order to perform the actions they desired. Some of the features added in this prototype were not used, such as Orghz and Human Flying Units production or the Upgrades available on the Research Building Structure. Still most of them could be observed even if sporadically. If there was the possibility on having the Users that participated on these tests, being available to perform more sets of tests using this prototype, maybe they would choose the new features present on it more often, than the basic units and structures available.

7. Conclusions

Reviewing the goal of this thesis: Designing and Creating a RTS game, using HTLM5 and Javascript programming languages, that can be played on both personal computers and mobile devices, containing the traditional 4X elements present in this game genre and being played in multiplayer mode. By reading this document it was proved that this goal was accomplished.

To do so various tasks had to be applied, since making a RTS involved various academic strands in order to create an implement a game of this magnitude. There was the game design that was conceived for this game, carefully designing each aspect that is part of it such as Units, Structures, Economy, Upgrades, 4X, Super Power and others. Then it was presented the Architecture that is part of this game, referring the Technical Limitations present on this game, how it would be structured the communications between the Game Server and the Game clients and how the game project was organized with the use of the *Phaser* framework. For the implementation of this project there were implemented several tasks presented on this document, who contributed the most to fulfill the main objectives present in this thesis.

For being able to observe if the objectives of this thesis were accomplished, User Tests were conducted with the Focus Group chosen for this project. They were done in order to maximize the game experience of *Planetary Conquest*; see the usage of some elements that were part of this project's gameplay such as the balance between the Units, Structures and Monster attributes; observe if the interface could be understood by the participant players and apply the game actions they desired with the help of the game controls; and what were their emotions when they were playing the game. From the two sets of tests performed, it was observed on the **Alpha prototype** that it was possible to play a multiplayer RTS game with the main objectives present on this thesis on computer platforms, with the users understanding which actions they had to perform in order to defeat their opponents. From the **Beta prototype** it was observed that *Planetary Conquest* can be played on computer platforms and mobile devices. Also most of the features present in this prototype were experimented by the User Testers.

In spite of some features not being implemented such as Fog of War and Collision Avoidance between moving game object, it was possible to create a multiplayer RTS game that can be played on multiple game platforms, with the simple use of a web browser.

For the future work purposed for this project it would be good to review some of the errors made on it, observe what it can be done to the tasks that could not be implemented, and implement some other tasks that could enhance the game experience. Some of those tasks include adding game sounds to this project, adding the ability to have 2 vs 2 player matches and having the game server capable of having authority over multiple game matches being played at the same time.

References

[1] Essencial facts about the computer and video game industry. Entertainment Software Association, 2015.

[2] E. Adams. *In Fundamentals of Game Design*. New Riders, 2010.

[3] Z. D. Boren. There are officially more mobile devices than people in the world. Independent, 2014.

[4] W. Burrows. *In Grand Street 37*. W W Norton & Co Inc, 1991.

[5] P. S. Carlos Martinho and R. Prada. *Design e Desenvolvimento de Jogos*. FCA, 2014.

[6] A. Emerich. Microprose's strategic space opera is rated xxxx. Computer Gaming World (Issue #110), 1993.

[7] D. Flannagan. *JavaScript - The definitive guide*. O'Reilly Media, 6th edition, 2011.

[8] D. Kosak. Top ten real-time games of all the time, 2004.

[9] A. Sartori-Angus. Cosmic conquest. *BYTE*, 1982.

[10] A. R. Shankar. *lock-step method*. Apress, 2th edition, 2017.

[11] R. Silveira. Multiplayer game programming. 2015.