Multiplier
Real-Time Strategy Unit Balancing Tool

By Thompson Lee

A Project Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirement for the

Degree of Master of Science

in

Interactive Media & Game Development

May 2016

Advisor: Professor Dean O'Donnell

Reader: Professor Brian Moriarty

Reader: Professor Charles Rich

Table of Contents

Table of Figures 3
Abstract 4
Acknowledgements 4
1. Introduction 5
1.1. Terminology of Real-Time Abstract Strategy 5
1.2. The Differences: Real-Time Strategy vs. Real-Time Tactics 5
1.3. Origins of the Custom-Built Software and How It Ties in to RTAS, RTS, and RTT 7
1.4. Game Balance and Methods to Achieve Balance 8
1.5. Game Feature and Appeal 14
2. Game Design 14
2.1. Overview 14
2.2. History 15
2.3. Game Mechanics 16
2.3.1. Attributes Editor 16
2.3.2. Splitting and Merging 17
2.3.3. Minimap 18
2.3.4. Autonomous Attacking 19
2.3.5. Lack of Fog-of-War 19
2.3.6. A.I. Gameplay 20
2.3.7. Local Online Multiplayer 20
2.4. How Unit Attributes Relate to Game Design 21
3. Resources 22
4. Evaluation 23
4.1. Experimental Setup 23
4.2. Hypothesis 23
4.3. Result 25
4.3.1. Likert Scale Questions 25
4.3.2. Mean of Likert Scale Questions 26
4.3.3. Open-Ended Questions 27
5. Conclusion 33
6. Future Work 34
7. Postmortem 34

7.1. What went right? 34
7.1.1. Choosing Unity 34
7.1.2. Scoping the Project 35
7.2. What went wrong? 35
7.2.1. Tutorial Manager 35
7.2.2. Multiplayer and “New Multiplayer” 37
7.2.3. Post-Test Survey Could Have Been Better 37
7.2.4. Being a Sole Developer 38
7.3 What did I learn? 38
7.3.1. Things do not always go the way you want them to go 38
7.3.2. Never Use Planned Deprecated Software 39
8. References 39
9. Appendices 44
9.1. Post-Test Survey 45
9.2. Project Journal 47
9.3. Original Master’s Project Proposal (OLD) 51
References 56

Table of Figures

Figure 1: Agar.io(left) and Auralux(right). 6

Figure 2: An example of three different levels of veterancy, shown by the number of small white bars above their blue health bar. 7

Figure 3: Total War: Shogun 2 gameplay. 8

Figure 4: Total War: Attila gameplay. 9

Figure 5: Multiwinia gameplay. 10

Figure 6: StarCraft II gameplay. 11

Figure 7: WarCraft III: The Frozen Throne gameplay. 12

Figure 8: Conquest: Frontier Wars gameplay. 12

Figure 9: Footman Wars map gameplay. It is a custom symmetrical map played in WarCraft III: The Frozen Throne. 13

Figure 10: Galcon gameplay. 13

Figure 11: Multiplier, the real-time strategy unit balancing tool, a.k.a., the "Software - Tool version". 15

Figure 12: The Attributes Editor for Single Player Mode in the Tool version, with the Tooltip and the corresponding indication box shown. 17

Figure 13: Splitting (left) and Merging (right). 18

Figure 14: Minimap and camera boundaries. Note: Text cannot be removed in Unity 5 Education and will always be present. 18

Figure 15: The minimap shown without any fog-of-war. Some of the units are taking damage and attacking opponents. 19

Figure 16: Simulation Mode shown in the Tool build. 20

Figure 17: Multiplayer Mode in the Tool version, showing the LAN options, Attributes Editor and the Level Difficulty settings. The editor will only appear when the player chooses the "Custom" preset option. 21

Figure 18: Table of attributes and applications. 21

Figure 19: Table of open-ended questions in the post-test survey form. 24

Figure 20: Graphical interpretation of the results of the 6 5-point Likert scale questions from the post-test survey. 25

Figure 21: Mean of Likert Scale Points ordered by Questions and Group, with 95% confidence interval. 26

Figure 22: Graph of the number of participants answering the open ended question related to Appeal. 27

Figure 23: Graph of the number of participants answering questions about effectiveness of software features and other aspects. 28

Figure 24: Graph showing the number of participants requesting features to keep and features to improve. 29

Figure 25: Graph showing the numbers of participants rating the best and worst features from both groups. 30

FIGURE 26: GRAPH SHOWING THE NUMBER OF PARTICIPANTS WILLING TO CONTINUE PLAYING THE SOFTWARE AT ITS CURRENT AND IMPROVED STATE. 31
FIGURE 27: GRAPH SHOWING THE NUMBER OF PARTICIPANTS RATING THE DIFFICULTY OF USING THE SOFTWARE.....31
FIGURE 28: GRAPH SHOWING HOW MANY PARTICIPANTS ARE CONFUSED OF CERTAIN FEATURES. 32
FIGURE 29: GRAPH SHOWING PLAYER FEEDBACK, CATEGORIZED INTO MANY DIFFERENT SECTIONS. 32

Abstract

We have built an application that integrates a technical editor feature and a custom real-time strategy game. The end users are able to use the technical editor feature for tweaking and customizing the unit attributes and progressions in the game using simple mathematical formulas, and they can play or test their tweaked formulas within the game. Various game modes in the software, which are Single Player, Multiplayer, and Simulation, can help display to the end users the results of their tweaked formulas, or users can just have fun by playing the game.

The software was evaluated to see whether the software with the editor feature enabled is more attractive and appealing to the end users than the software with the editor feature disabled. The evaluation is based on the players' feedback on the game with or without the editor. A total of 50 testers were randomly assigned into 2 groups evenly, the Tool group and the Game group. Testers assigned to the Tool group were able to customize the game unit attributes via the editor and play, while the testers in the Game group only play the game.

The results from the post-test survey show both versions of the software are highly appealing to the testers, and there is no significant difference in game appeals between the Tool version and the Game version.

Acknowledgements

I would like to thank Dean O'Donnell as supervisor for providing guidance, Brian Moriarty and Charles Rich as readers for providing assistance, and Jennifer deWinter for feedback from a non-technical perspective.

I would like to thank the volunteers and testers for their feedback. Their feedback helped to improve the game, and made the game as it is.

I would also like to thank the Unity3D and Reddit communities for providing assistance, giving advice for improvements, and the encouragement for completing the assignment.

1. Introduction

1.1. Terminology of Real-Time Abstract Strategy

"Real-time strategy," is defined as a strategy video game genre in which the game does not progress incrementally in turns, i.e., all actions are simultaneously executed.[1] These games require the players to do real-time planning based on incomplete information of their surroundings and they need to handle resource management themselves.[2] Resource management refers to the management of available resources the players have access to over the course of the game.

Abstract strategy games are games with minimal to no themes. The word, "abstract" refers to games without having no themes, or having themes that are not important to the experience of playing.[3]

"Real-time abstract strategy" (RTAS) is a new subgenre of abstract strategy games that are played in real-time. Example games which show such traits include *Agar.io*[4] and *Auralux*[5]. The nature of games where the actions are played out simultaneously is synonymous with "real-time", thusly the term "real-time abstract strategy" is defined to be a subgenre of "abstract strategy", mixed with elements of "real-time strategy" gameplay.

1.2. The Differences: Real-Time Strategy vs. Real-Time Tactics

Real-time strategy games are sometimes confused with real-time tactics (RTT), in terms of game mechanics and gameplay.[6] However, they both carry the characteristics of real-time gameplay.

In real-time strategy games, players devise intricate strategies involving collections of resources, base-building, technology upgrades, and unit types to take advantage of what they believe their opponents will do, and what strategies their opponents will use, without any prior knowledge.[7]

Base building represents the dynamics of players securing the locations of abundant resources by base expansions. Unit building represents the limited selection of units that are available to produce at any one time. Macromanagement represents the general economy aspects of managing the intake and expenses of the player's resources, such as constructing buildings, conducting research and technology upgrades, and the purchase of unique units and items affecting overall gameplay strategies.[8]

[1] (Xiong & Iida, 2014)
[2] (Cheng & Thawonmas, 2004)
[3] (Thompson, 2000)
[4] (Miniclip, 2016)
[5] (War Drum Studios, 2016)
[6] (Walker, 2004)
[7] (Kleinberg, 2011)
[8] (Giant Bomb, 2016). Macromanagement is derived from micromanagement in real-time strategy games.

Figure 1: Agar.io[9](left) and Auralux[10](right).

These strategies usually involve choosing what unit groups the players will be using among the many available types of units and types of upgrades in the game. Players can apply upgrades for their units to perform better than they would expect the performances of their opponents' units, or playing mind games to deceptively lure their opponents to their downfall.[11] As the game progresses, these player decisions will affect the overall outcome of the game. In other words, the master plan of all actions done will determine who wins the game.

Real-time tactics is a subgenre or a related genre of real-time strategy games, with the aspects of base building, unit building and the importance of macromanagement all removed from the game. Instead, it is about the placements of units on the battlefields, the unit troop formations, and the exploitation of terrain, the environment, and territory acquisition for tactical advantages against enemies. In short, it is all about how you win each battle. Usually, players are provided with limited available resources, such as a given set of units provided in missions, and are tasked to complete game sessions using only those resources. Strategies to preserve limited resources, such as utilizing veterancy where units gain permanent bonuses when leveling up[12], through killing more enemy units or staying alive longer, thus becoming more effective[13].

[9] Agar.io screenshot taken from: http://agar.io/
[10] Auralux screenshot taken from: https://www.auraluxgame.com/game/
[11] (Lahiri, 2010)
[12] (YurdleTheTurtle, 2010)
[13] (Wikia, 2014), (Wikia, 2016), and (OpenRA, 2016). The Wikia references are taken from Internet Web Archives. OpenRA is an open-source reimplementation of Westwood Studio's Command and Conquer: Red Alert.

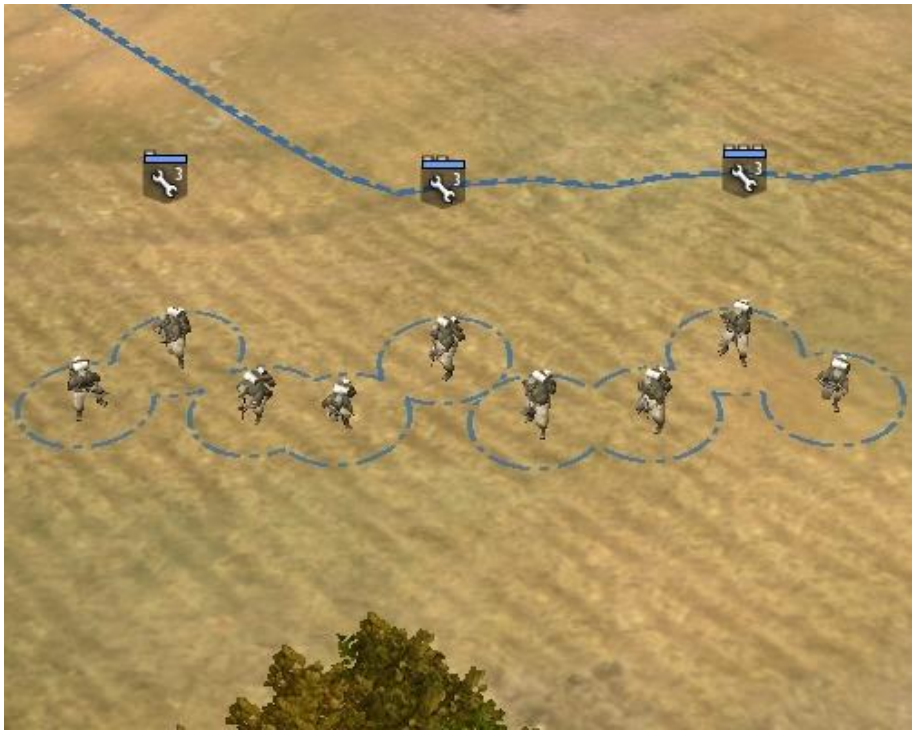

Figure 2: An example of three different levels of veterancy, shown by the number of small white bars above their blue health bar.[14]

1.3. Origins of the Custom-Built Software and How It Ties in to RTAS, RTS, and RTT

Originally, the idea was to come up with a way to see if integrating mathematical equations into a real-time strategy game would work. Let's say X_n is the increase of a unit upgraded from Level N to Level N+1.

$$X_n = f(\mathrm{n}+1) - f(\mathrm{n}) \qquad \text{Equation (1)}$$

If X_n is a sine wave mapped to the Y axis from 0 to 1 on the X axis, what would happen? Will the game balance be disrupted? The main focus of this idea was to experimenting with game balance using mathematical equations.

One proposed approach to working with mathematically tweakable game balance was to build a real-time strategy game with a tool allowing simple mathematical equations to determine usable unit's attributes, and assess the outcome. Note that this approach is not deemed to be the best approach of evaluating game balance through mathematical means. Nonetheless, if the results show that units with mathematically generated unit attributes can be used in a real-time strategy game and the game is deemed to be balanced to play after multiple counts of gameplay sessions, it may pave the way to future works in the realm of procedural content generation in real-time strategy unit design.

[14] Screenshot taken from GameReplay.org:
http://www.gamereplays.org/companyofheroes/portals.php?show=page&name=how-veterancy-works

Multiplier takes game elements from real-time abstract strategy, real-time strategy, and real-time tactics. It uses generic units, represented as simple, geometric shapes, and they increase in size while they level up as the game progresses. All unit attributes, including permanent bonuses each unit will have as they level up, are completely customizable to the player beforehand. This is how it incorporates unit building and veterancy into the gameplay.

1.4. Game Balance and Methods to Achieve Balance

Game balance is about implementing fairness in a game so that each side has an equal chance of winning by even play, handicap play, or odds play.[15] The developers try to level the field and use the game's environment and resources to determine this balance.

Figure 3: Total War: Shogun 2 gameplay.[16]

Real-time strategy games are notoriously known for their high difficulty when it comes to game balancing.[17] Players can choose amongst various factions and units with different strengths and weaknesses, and the developers must carefully test all potential interactions and ensure they are balanced and fair across different types of terrain, maps, game modes, and scenarios. There is a particular interesting concept, called the "Nash equilibrium", defined as "the existence of an equilibrium state where no players can benefit from

[15] (Pedersen, 2009)
[16] Screenshot from 3D Juegos: http://www.3djuegos.com/12579/shogun-2-total-war-la-caida-de-los-samurai/
[17] (Egenfeldt-Nielsen, Smith, & Tosca, 2012)

changing their strategies."[18] This means the players will tend to gravitate towards the most optimum strategy, or the dominant strategy in an unbalanced game. The existence of such dominant strategy saps away the potential for choice, and makes the game boring to play.[19] Therefore, game developers strive to avoid making their games reach the Nash equilibrium.

Figure 4: Total War: Attila gameplay.[20]

We look into methods of balancing games by having a couple of strategies for the players to choose and avoiding Nash equilibrium, in order to remove factors that may hinder the evaluations. We also look into the option of allowing players to balance their units, to see if this increases the potential choices of strategies the players can choose from, so the players will not be bored by the game and lose the game's appeal.

Games that have moderate dynamics and balancing may be used as references for designing Multiplier. *Total War: Shogun 2*[21], *Total War: Attila*[22], and *Multiwinia*[23] are all real-time strategy games where unit compositions on opposing teams are the same, and require the players to use strategic unit troop placements on the battlefield to win battles. In these games, the battlefield area is large enough to provide ample stimuli for players to venture out and prepare for battle.

[18] (Nash, 1950)
[19] (Egenfeldt-Nielsen, Smith, & Tosca, 2012)
[20] Screenshot from High-Def Digest: http://games.highdefdigest.com/18679/total_war_attila_pc.html
[21] (Onyett, Total War: Shogun 2 Review, 2011)
[22] (Hafer, 2015)
[23] (Griliopoulos, 2008)

Figure 5: Multiwinia gameplay.[24]

Games with asymmetrical gameplay, involving more complicated unit attributes, or geographical properties that affect player decisions, require more complex game balance, such as *Starcraft II*[25], *Warcraft III*[26], *Conquest: Frontier Wars*[27]. In these games, unit attributes are affected by dynamic properties of the units, such as speed, regeneration, and cooldowns, which are incrementally increased through researching tech upgrades. These bonuses added to the unit attributes were shown to determine the outcome of real-time strategy multiplayer game sessions[28], and thusly, makes them attractive to the players and motivate them to obtain these upgrades.

Other than unit attributes or geographical properties, unit types and compositions can drastically affect game balance. For example, in *StarCraft II*, you have the Terran Battlecruisers and the Protoss Carriers, both are armored air units. A Battlecruiser has a stronger attack, higher armor points, and shorter range, while a Carrier has a longer range, lower armor points, and has a weaker attack. Carriers are also

[24] Screenshot taken from NewGamerNation: http://www.newgamernation.com/review-multiwinia-survival-of-the-flattest/
[25] (Blizzard Entertainment, 2015)
[26] (Blizzard Entertainment, 2002)
[27] (Todd, 2001)
[28] (Bangay & Makin, 2013)

weak to Battlecruisers[29], and this weakness and the differences in unit attributes can make the players think about what units to build and produce, and by priority and planning.

Figure 6: StarCraft II gameplay.[30]

Similarly, some real-time strategy games, such as *Auralux*[31] and *Footmen Wars*[32], utilize map layouts that are designed symmetrically. *Auralux* provides the basis of linear upgrade paths that players can use during gameplay, as well as taking into account of the map layout. *Footmen Wars* provides a similar structure of gameplay, in which each unit of different factions have attributes that players can upgrade accordingly, but ultimately, the players can only use that unit for the rest of the game.

Research[33] has already been done on exploring map layouts and game balance in real-time strategy games, exploring this using *Planet Wars* with procedurally generated maps, which the game is based on *Galcon*[34], as the basis of their research. They used an evolutionary strategy with 4 parameters, X position, Y position, growth rate, and max capacity of ships, to generate maps with an arbitrary number of neutral planets ranged between 15 and 30, while following the rules of the game. After generating a set of playable maps and running 2 experiments with 10 executions using a tournament system to assess the quality of the maps, they found that as long as the map layout does not provide players more advantages over their opponents, regardless of the abilities or strategy type, it is considered to be a balanced map. These generated

[29] (Blizzard Entertainment, 2016)
[30] Screenshot taken from Sep7agon Gaming Forums: http://sep7agon.net/gaming/starcraft-2-review/
[31] A minimalistic real-time strategy game for Android, based in outer space. (Parker, 2013)
[32] A real-time tactics custom map game for the real-time strategy game, *WarCraft III* and its expansion, *WarCraft III: The Frozen Throne*. (StrategyWiki, 2014)
[33] (Lara-Cabrera, Cotta, & Fernández-Leiva, A Procedural Balanced Map Generator with Self-adaptive Complexity for the Real-Time Strategy Game Planet Wars, 2013)
[34] *Galcon* is a real-time strategy game where you conquer planets by invading with your ships. (Hassey, 2016)

maps make the game more interesting to weak players (as they cannot lose easily), and raising the competitiveness of strong players with harder challenges.

Figure 7: WarCraft III: The Frozen Throne gameplay.[35]

We can start to see many varieties of ways to approach to game balance in real-time strategy games, but most of all, unit interaction is one of many core components of real-time strategy games.[36] Experimenting the possibilities of game balancing using mathematical equations in the realms of unit interactions, therefore, becomes the main focus of this research project.

Figure 8: Conquest: Frontier Wars gameplay.[37]

[35] Screenshot taken from Giant Bomb: http://www.giantbomb.com/warcraft-iii-the-frozen-throne/3030-14073/
[36] Unit interaction is discussed in the Introduction of (Li Yan, 2014).
[37] Screenshot taken from ModDB: http://www.moddb.com/games/conquest-frontier-wars1/downloads/conquest-frontier-wars-patch-107

Figure 9: Footman Wars map gameplay.[38] *It is a custom symmetrical map played in WarCraft III: The Frozen Throne.*

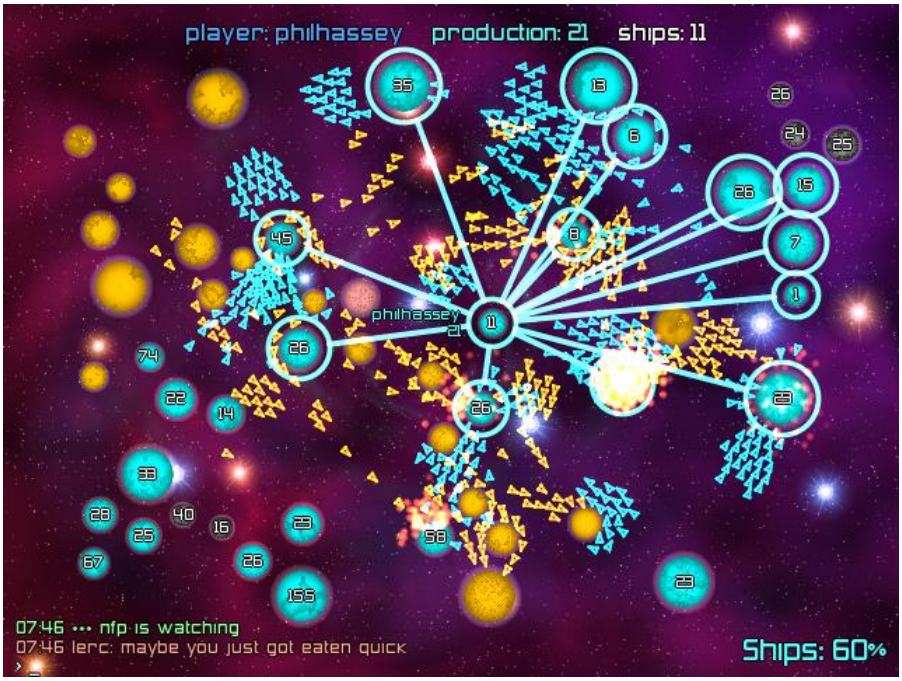

Figure 10: Galcon gameplay.[39]

[38] Screenshot taken from Gaming Tools: http://gaming-tools.com/warcraft-3/footmen-frenzy/
[39] Screenshot taken from Galcon Games: http://www.galcon.com/games/?action=game&name=galcon

1.5. Game Feature and Appeal

Since game balance itself is a very debatable game design aspect in real-time strategy games, and due to how ambiguous the decisions of approaches for tweaking the game are, the impact of being able to modify unit attributes may be a type of experience the players like. We therefore approach this by creating a game with this aspect as a game feature, and conducting a survey to see how well it is received.

2. Game Design

2.1. Overview

In this section, the game players and the tool users are all described as "end users", and both the version of Multiplier with the editor enabled, called "Tool version," and the version of Multiplier without the editor, called "Game version," are referred to as "software". The software is built using the latest Unity3D Game Engine, Unity 5 Education.[40]

We intended the editor of the software to be simple enough for end users to use easily. We figured if the editor is easy to use, the end users will be able to utilize it in a variety of ways. This means "the usage of the editor" itself becomes its own rules of play. We call this, "metaplay," which is a different approach of playing games, where the player is no longer playing the game as it was designed or intended by the creators, but rather changing the rules of the game around, exploring its limitations, creating and pursuing personal objectives, and other motivations.[41] We wanted to incorporate the metaplay as a gameplay element of our software, and evaluate its game appeal compared to just the game.

The editor interacts with the software and tweaks the properties of the given 6 unit attributes for each player's units used in the game, each player being the end users and the game A.I. player. The editor also allows the end users to define any possible leveling progressions, or power-ups, using mathematical equations, including making the power-ups progress towards the negatives if the end users wished so. The intent is to give the end users a lot of freedom to set any unit attribute values, called "level rates," to the units, and that includes setting values to the negatives.

The gameplay of the software is a real-time abstract strategy game, purposefully built in a way where game units split and merge themselves to grow or "upgrade". The goal of the game is to wipe out the opponent's units to win. The end users split their units to create more resources, and use the extra units to merge. When merging, units will upgrade their units to the next level, at the expense of a unit of the same level prior to merging. As the end users continue to split and merge units, they will reach a point where neither player will win, or will win after certainly a long period of time.

[40] (Unity3D Technologies, 2016)
[41] (Breslin, 2009)

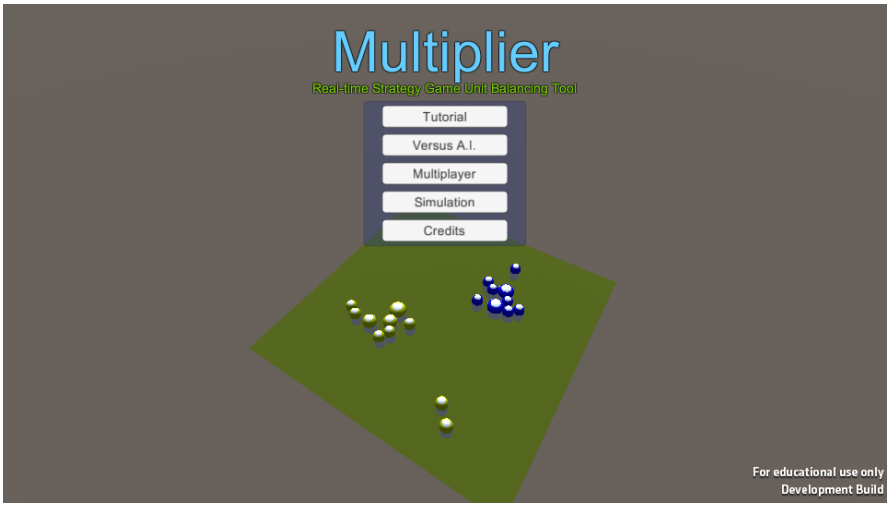

Figure 11: Multiplier, the real-time strategy unit balancing tool, a.k.a., the "Software - Tool version".

The software contains a variety of game modes, which are Single Player, Multiplayer, and Simulation. The Simulation Mode is a game mode where the end users are observing and simulating a battle between 2 game A.I players. This game mode allows the end users to test and tweak game units accordingly for each player. Only the tool version software build allows the end users to test and tweak the unit attributes, and help the end users verify if the game units are balanced enough. If the units are balanced, the end users may continue to use the tool version and choose to apply mathematical equations to their own games as their heuristics for a balanced unit leveling progression, or play against others in Multiplayer Mode. For the game version software build, the end users are able to play Single Player and Multiplayer modes, with Simulation mode left out as it is part of the editor. With the exceptions of the editor and Simulation mode, both software builds are identical.

2.2. History

The original premise of the software was a game designed around the possibility that complex unit interactions are defined using mathematical equations. Starting from very simple mathematical equations is a good starting point, so as not to be burden with how complex the mathematical representation was going to be, as well as the technical limitations to accomplish this. There were other considerations made while planning out the premise, even once suggested whether to venture forth into advanced generations of units whose interactions and relations are procedurally generated, but the scope of the game and the project itself forbid this.

When thinking about the composition of a real-time strategy game, it must contain a few elements that define the genre: simultaneous gameplay, limited time to execute decisions, and the complexity of the game in terms of the large number of actions available per decision cycle.[42] From a general point of view, defining elements are: resource management, base building, and enemy annihilation.[43] Optional elements

[42] (Ontañón, et al., 2013)
[43] These gameplay elements are observed from the many samples of real-time strategy games that are referenced. Note that not every real-time strategy game fits these criteria, but at least the majority of games do.

include stressing the importance of micromanagement and macromanagement, complicated unit interactions, and tactical strategies players can choose to put in practice. [44] All of these elements mean, the final game would have to incorporate common elements, and use certain game mechanics to satisfy them.

The inspiration of having basic units be upgraded to stronger units of the same borrows from real-time tactics games, in the same veins as *Footmen Wars*[45], where it is easier to reuse the same unit, but given stat boosts for upgrades.

To find the most simplistic math equation, the easiest solution constructed is to double up the number, or by doubling the result.

$$y = 2x, x \in \mathbb{N} \qquad \text{Equation (2)}$$

Using the above equation, game units would be exactly twice more powerful when upgraded, and continue to be exactly twice as powerful for subsequent upgrades. It also makes designing a real-time strategy game easier to conceive, but harder to expand upon for flexibility. The next solution is to come up with some new equations that are still simple to remember, but add a bit of complexity. These equations are given as follows:

$$y = x_n + x_{n+1}, x_0 = 1, x_1 = 1, n \geq 0 \qquad \text{Equation (3)}$$

$$y_n = x - y_{n-1}, n \geq 0 \qquad \text{Equation (4)}$$

It forced upon the idea that math equations should not be limited to just doubling up the results, and should use commonly known math operators to create complex results. And to support this, comes with a UI that handles these. The addition of the UI changed the software focus of being a game to a tool.

2.3. Game Mechanics

2.3.1. Attributes Editor

The editor (see Figure 12) consists of many UI components that provide the end users the ability to tweak the numbers and values at a glance.

Starting from the top of the editor and moving down, the editor is displayed with Player Configurations and A.I Player Configurations. In the Player Configurations, it has Presets, Category, Leveling Rates, and the Custom Equation. Presets is a dropdown menu where the end users can choose preset configurations available. Category is a list of unit attributes the game supports, and the end users can set which unit attribute to modify by choosing a unit attribute. Leveling Rates are panels indicating the level rates of the chosen unit attribute for that level. The top text tells the player the level number the level rate applies to. The center text is the level rate. And the bottom text shows if the next level rate is greater or less than the current level rate. Finally, the Custom Equation is a text field the end user can type equations in, and the results from Level 1 to Level 10 are shown in the Leveling Rates panels. All of the UI components in the Player Configurations are the same in A.I. Player Configurations.

[44] Most real-time strategy games come with campaign modes, which utilizes these optional elements. However, depending on the gameplay experienced in multiplayer skirmishes, these elements may not appear dominantly.
[45] (StrategyWiki, 2014)

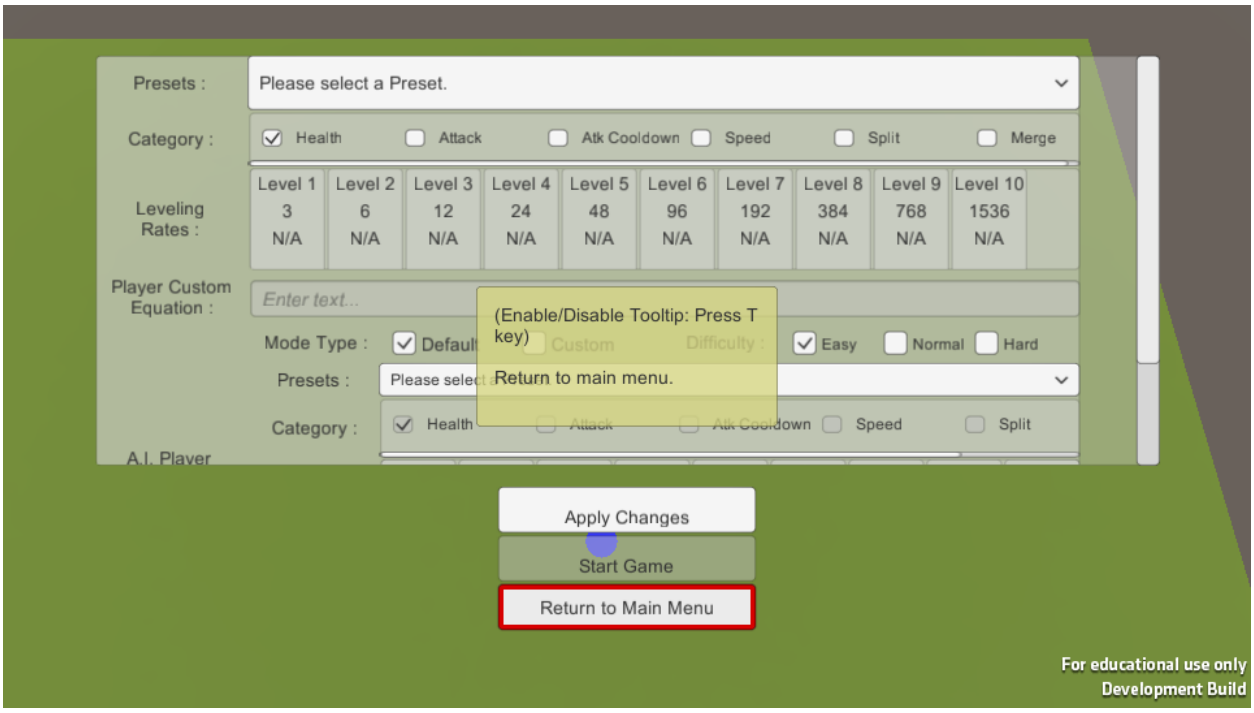

Figure 12: The Attributes Editor for Single Player Mode in the Tool version, with the Tooltip and the corresponding indication box shown.

In the A.I. Player Configurations, there is an extra UI component called the Mode Type. Mode Type allows the end user to use the default, preconfigured A.I. player configurations, or customize the configurations by themselves. They can also choose the difficulty level of the A.I. player.

The end users can enter in a math equation in the Custom Equation. Using the Shunting Yard Algorithm[46], the provided equation is then parsed and calculated to correctly print out the results and are shown for each level in the Leveling Rates, reaching up to Level 10.

Due to the many UI components possibly confusing the end users, toggleable tooltips (yellow text box in Figure 14) have been applied everywhere to provide hints and tips on how to use them to the end users.

2.3.2. Splitting and Merging

The main game mechanics are game unit splitting and merging (see Figure 13). Splitting is where the unit creates a duplicate of itself, and can sometimes be described as cell mitosis in biology. Merging is where a pair of units of the same level merges, creating 1 upgraded unit, scaled slightly larger than its previous size. The merging process can be described as cell fusion in biology. Splitting creates more units for the player to merge, starting from the most basic level (Level 1), and merging up to the highest level (Level 10).

[46] (Dijkstra, 1961)

Figure 13: Splitting (left) and Merging (right).

Both of these unit abilities can be said as confusing to grasp, since they are not very common game mechanics used together within the game context. Depending on the given math equations that define the traits of the unit, or unit attributes, upgraded units may not be positively stronger as other units of different levels in comparison. For example, using Equation (3), it shows for every 2 merges (upgrades), the unit attributes increment once. This means, Level 2 units are the same as Level 1 units, but Level 3 units are stronger than Level 2 units.

2.3.3. Minimap

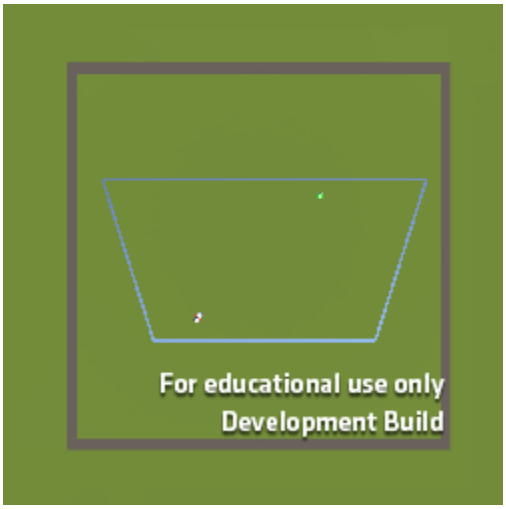

Figure 14: Minimap and camera boundaries. Note: Text cannot be removed in Unity 5 Education and will always be present.

Inspired by the camera panning movement game mechanics for the minimap from *WarCraft III*[47] and originating from *Dune II: The Building of a Dynasty*[48], the camera moves in the X and Z axes when the end users start dragging within the boxed region, allowing it to move freely around on the map. This is added into the game to allow the end users to quickly see where the actions are occurring in the map level. This minimap also allows the end users to precisely command their units to move to a location where it would mostly be out of bounds in the main camera.

2.3.4. Autonomous Attacking

When enemy units are nearby, the end user's units will go chase after them and attack the enemies once they are near enough. This helps to prevent the end user from solely focusing on micromanaging the game units, and to observe the unit interactions with the tweaked unit attributes.

2.3.5. Lack of Fog-of-War

Part of our goal is to give all players complete information of the game by designing it as a real-time abstract strategy game. Fog-of-war[49], is known in the military, referring to the general friction of forces, which are danger, exertion, uncertainty, and chance, comprising the climate of war.

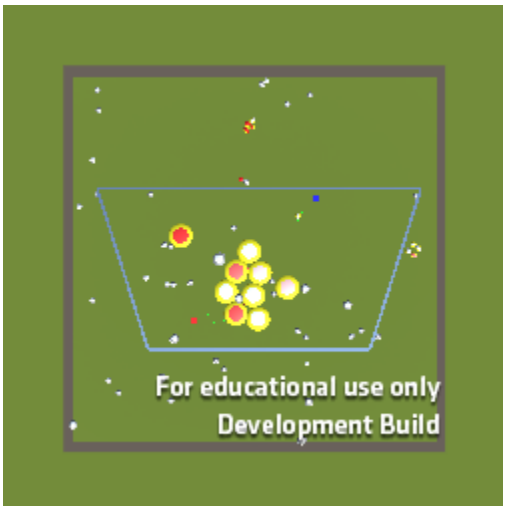

Figure 15: The minimap shown without any fog-of-war. Some of the units are taking damage and attacking opponents.

In typical real-time strategy games, fog-of-war refers to the black fog covering the world in pitch dark unless revealed to the player when their units are nearby, simulating these general friction of forces.

[47] (Blizzard Entertainment, 2002)
[48] (Devkar, 2003)
[49] (Shepherd III, 1997). This paper also goes on to define what the term, "fog of war," is about.

In other words, only the players can see objects within line of sight of their units. Players can only guess information retrieved from the glimpse of visible portions in the scene and plan out their strategies using known information. This software lacks such fog-of-war element, so that the end users are expected to look at unit interactions and to observe if their units are balanced.

2.3.6. A.I. Gameplay

This feature can easily be observed in Simulation Mode, and is playable in Single Player Mode. The game A.I. uses finite state machines, with each finite state machine carrying out a specific order based on the game A.I.'s status. Which units to carry out the commands, and where the units are to go to, are all determined randomly. The game A.I. also utilizes trick merges heavily, in which 2 units on the map very far apart from each other merges, creating an upgraded unit in the middle.

Figure 16: Simulation Mode shown in the Tool build.

These features of gameplay are what make up the entire software, allowing the end users to use or play them whichever they want.

2.3.7. Local Online Multiplayer

We also provided Multiplayer Mode, a game feature where the end users can host or join an online hosted local multiplayer game. This feature is commonly implemented in both versions of the software. The game is not just about testing unit balance against a game A.I. player, but also testing unit

Figure 17: Multiplayer Mode in the Tool version, showing the LAN options, Attributes Editor and the Level Difficulty settings. The editor will only appear when the player chooses the "Custom" preset option.

balance against a human player. This mode allows the end users to test certain scenarios that will only occur in a human vs. human game match session, usually in competitions and ladder/skirmish matches, custom maps, or metaplaying.

The host and client players are able to choose whichever mathematical equations they want to use when playing. In the Tool version, we provided the Attributes Editor to each of the players by selecting the Custom option, while we only provided Level Difficulty and no Custom option in the Game version.

2.4. How Unit Attributes Relate to Game Design

To explain, the 6 main unit attributes used in the software are applicable to most types of gameplay. They are built upon game elements from the mechanics commonly seen in real-time strategy games to form correlations that apply. The following tables show what each unit attribute can be applied to:

Attributes	**Game Mechanics**
Health	Unit health dependent of type, moveable buildings
Attack	Damage power, damage reduced by armor using nerfs/buffs percentages
Speed	Unit movement speed, traversal speed, speed-affecting unit type
Attack Cooldown	Pause interval between attacks, cooldown-reducing buffs, boss attacks
Split	The time players spend on producing and obtaining resources
Merge	The time players spend on upgrading, manufacturing, constructing, leveling

Figure 18: Table of attributes and applications.

- Health typically applies to unit health, health of a certain type of units, or it could be movable buildings.
- Attack would usually refer to how much damage the unit can deal to another unit, how much damage reduced caused by armor upgrades (by multiplying the attack reduction on the armor or attack stats upgrades, or multiplying the reciprocal of armor upgrades, attack-weakening buffs).
- Speed is often referred to as the speed of the unit's movement, or traversal speed, or how light/heavy the units are.
- Attack Cooldown may refer to the short pause between each unit attack, the buffs of obtaining an item that reduces cooldown, or the intervals of boss attacks.
- Split may refer to the time the players has spent on in producing, gathering, and obtaining resources.
- Merge may refer to the time players has spent on tech upgrades, tier leveling, construction of resources, and so on.

Of note, other unit properties, such as mana, health regeneration, collectible weapons, unit scale size, and area-of-effect status changes, are not being considered due to the scope of the project.

There are many other potential usages in which the 6 main unit attributes can apply to, and it doesn't have to be very restrictive. Each of these potential usage is therefore related to game design, even though their representations may be different.

3. Resources

The main resources used is the Unity game engine documentation, online tutorials, and first-hand sources for using Unity Networking, or UNET for short. Bugs/glitches tends to pop up from time to time, but we assessed that UNET is very reliable when it comes to internet connection stability. The Unity3D engine developers mentioned they will improve UNET further down the line as more fixes coming soon.

Other resources include the following:

- Github, the main repository hosting site where the project is hosted.
- Unity3D Forums, the main where professionals, amateurs, students, and hobbyists come and enjoy discussions on various matters. Usually, it is a central hub for technical questions and issues reporting to the Unity3D engine developers / staffs.
- Reddit /r/gamedev, a place where it is mostly showing off concepts of ideas that the game, or me, can witness and discuss about. It is also a big place where professional game developers come and discuss on the current trends, showcase their creations, or ask questions and answers them.
- Freenode IRC, an online internet relay chat channel where users can chat with other developers working on other things and helping each other out. Casual discussion and banter reside here, and is the author's main stress relieving go-to destination.

This project started from the summer of 2015, after the time where Unity3D had announced the release of Personal and Professional editions to all developers. It was not until the release of Unity 5.1.0f1, that initial work began. Due to technical issues, the project was completely overhauled upon the release of Unity 5.2.0, and work continued until now, as of writing.

4. Evaluation

This study is conducted to see if having a technical game feature in a game makes more players like the game more.

4.1. Experimental Setup

Once we had two versions of the software created, we want to assess the game appeal for both software. The only difference between the two versions is the editor, which is a technical game feature. We figured that having separate groups of testers, they can test out the two versions and assess what differences each version would be. These testers are then given a set of post-test survey questionnaires they each need to fill out, and we gather the data and compare the results.

There were a total of 51 participants involved in this experiment. One of the participants did not complete the questionnaire and is therefore excluded from evaluation. The rest of the participants are then split into 2 groups evenly, the Tool group and the Game group.

This study uses 2 versions of the custom-built software. One version, which has the editor feature enabled, is given to the Tool group to play around with. The other version, which has the editor feature disabled completely, is given to the Game group. Once both groups have completed their session, they are then given the post-test survey (see Appendix 9.1).

Before the participants start their testing session, they were given brief introductions on the control schemes and what the software is all about. Then they are left to try the software out themselves. Instructional answers were given to questions from testers arising from the testing session. A discussion of their feedback is written in this paper.

4.2. Hypothesis

We hypothesized that by having an extra technical feature added into the game, that game will have a higher game appeal.

For game appeal, we measured the appeal of metaplay and the gameplay for the Tool group and Game group, respectively. The use of this software will allow us to tell if incorporating the editor feature has more game appeal, compared with the standalone game. This study does not consider anything else regardless of how well the software is designed, implemented, and/or the level of polish. For preferences, we are measuring if the user prefers similar types of software built with editor features enabled or disabled.

This survey contains 6 questions based on a 5-point Likert scale, and 14 open-ended questions, mainly for feedback purposes. The 5-point Likert scale questions are based on a rated scale from 1 to 5, where 3 is neutral, 1 is in disagreement, and 5 is in agreement. Anything above 3.0 is considered as positive agreement. Anything below 3.0 is considered as negative agreement.

In the 6 5-point Likert scale questions, we asked the participants about the software's appeal, understanding of it, the likelihood of engagement in multiplayer, the replay value or replayability, fun, and preferences for similar types of this software. The questions and the categories associated with them are as follows:

1. This software appeals to me. (Appeal)
2. The software itself is easy to understand. (Understanding)
3. I like to play against someone else. (Likelihood of Engagement)
4. I want to continue playing this software. (Replayability)
5. This software is fun to use. (Fun)
6. I want to see more of this type of software. (Preference)

The open-ended questions are for determining what improvements the custom-built software may need to minimalize affecting factors. The answers given from the participants were used to track common issues, by measuring the frequency of common issues during the software evaluation process, and point out the trends that are common among the participants. These questions can be further categorized into larger grouped categories that may hint what aspects of the software need more improvements, as well as reflect upon other aspects in which would have affected the test results.

They are grouped as shown in the table below:

Category	Question
Appeal	Q3: Did the software consistently hold your interest?
Features	Q5: Did any aspects of the software stand out as particularly effective? How particularly ineffective?
	Q10: What features of the software do you wish to keep? List as many as you can.
	Q11: What features of the software do you wish not to keep? List as many as you can.
Preferences	Q8: What features of the software do you like the best? And the worst?
Replayability	Q12: Do you wished to continue using the software as it is right now?
	Q13: Do you wished to continue using the software once it is improved even more?
Understanding	Q1: What are the rules and objectives of the software? How did you discover them?
	Q2: How difficult is the software?
	Q4: Did anything about the software seem confusing or obscure?
	Q6: What would it have been good to know about the software before you started playing?
	Q7: How would you describe the software to a friend who has never used it?
Future Work	Q9: What potential aspects do you see about the software? List as many as you can.
	Q14: As a lucky beta tester, do you have any additional comments or questions that you would like to share?

Figure 19: Table of open-ended questions in the post-test survey form.

4.3. Result

4.3.1. Likert Scale Questions

The Likert scale points are ordered in descending order from 5 at the top to 1 at the bottom. If 1 is not shown, it means there are 0 results for 1. The following graph on the next page shows the majority of the participants rated positively to the questions.

Participants in both groups that highly agreed and agreed across these 6 5-point Likert scaled questions are marked as green for 5 and yellow for 4, respectively. Few participants that strongly disagreed or disagreed across the 6 questions are marked as red for 1, and blue for 2, respectively.

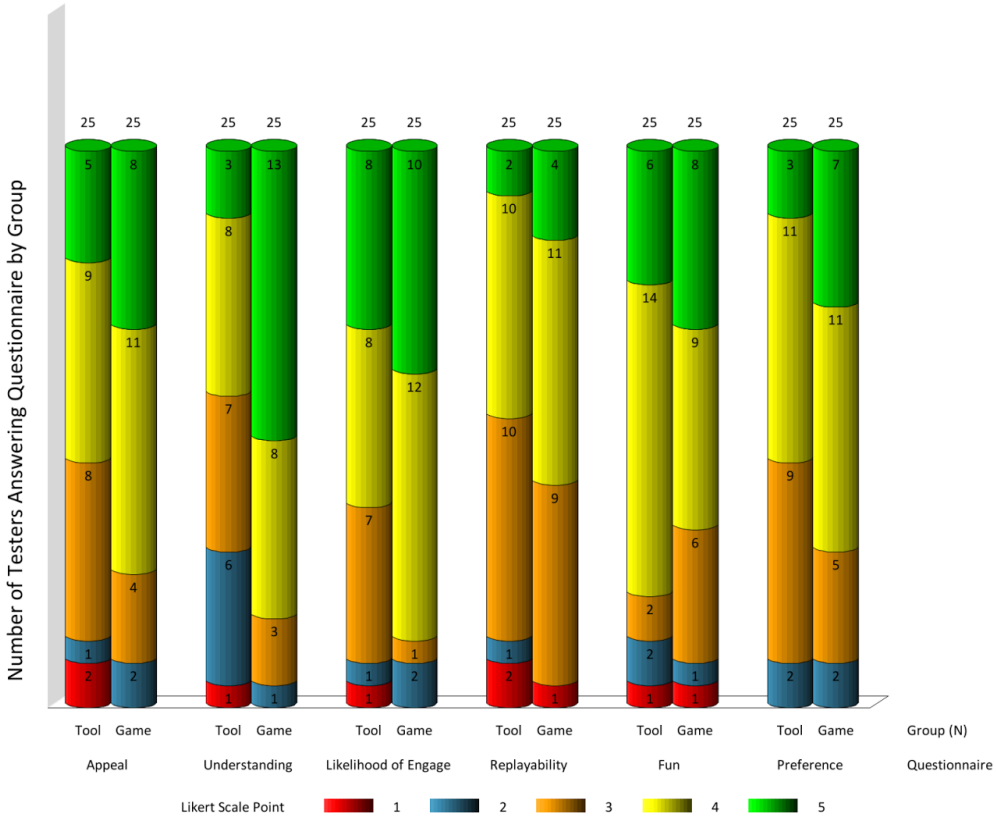

Figure 20: Graphical interpretation of the results of the 6 5-point Likert scale questions from the post-test survey.

Out of these 25 participants in the Tool group, 14 of them agreed the software is appealing, 7 of them disagreed that the software is easy to understand, 16 of them agreed they are more likely to engage in multiplayer, 12 of them agreed they will try the software again, 20 of them felt the software is fun, and 14 of them prefers similar types of software.

4.3.2. Mean of Likert Scale Questions

Out of these 25 participants in the Game group, 19 of them agreed the software is appealing, 21 of them agreed the software is easy to understand, 22 of them are more likely to engage in multiplayer, 15 of them are more likely to use the software again, 17 of them agreed the software is fun, and 18 of them preferred this type of software.

In the next following graph on the next page shown, we took the post-test survey results and analyzed them, which gives us the mean for each category.

The average is 3 on the 5-point Likert scale. The mean value for each question are above average for each group. Of interest, the p-value of the Understanding category shows that there is a significant difference in comparison between the Tool group and the Game group. The participants in the Game group agreed the software itself is easy to understand, while the Tool group had difficulty in understanding the software.

The p-value of the Fun category shows the participants in both groups strongly agreed the software is fun to play. There are no significant differences between the two groups for the other categories. It is interesting that even though the participants in the Tool group agreed the software appeals to them, it is fun to play, and they want to continue playing with it, they did not think the software is easy to understand.

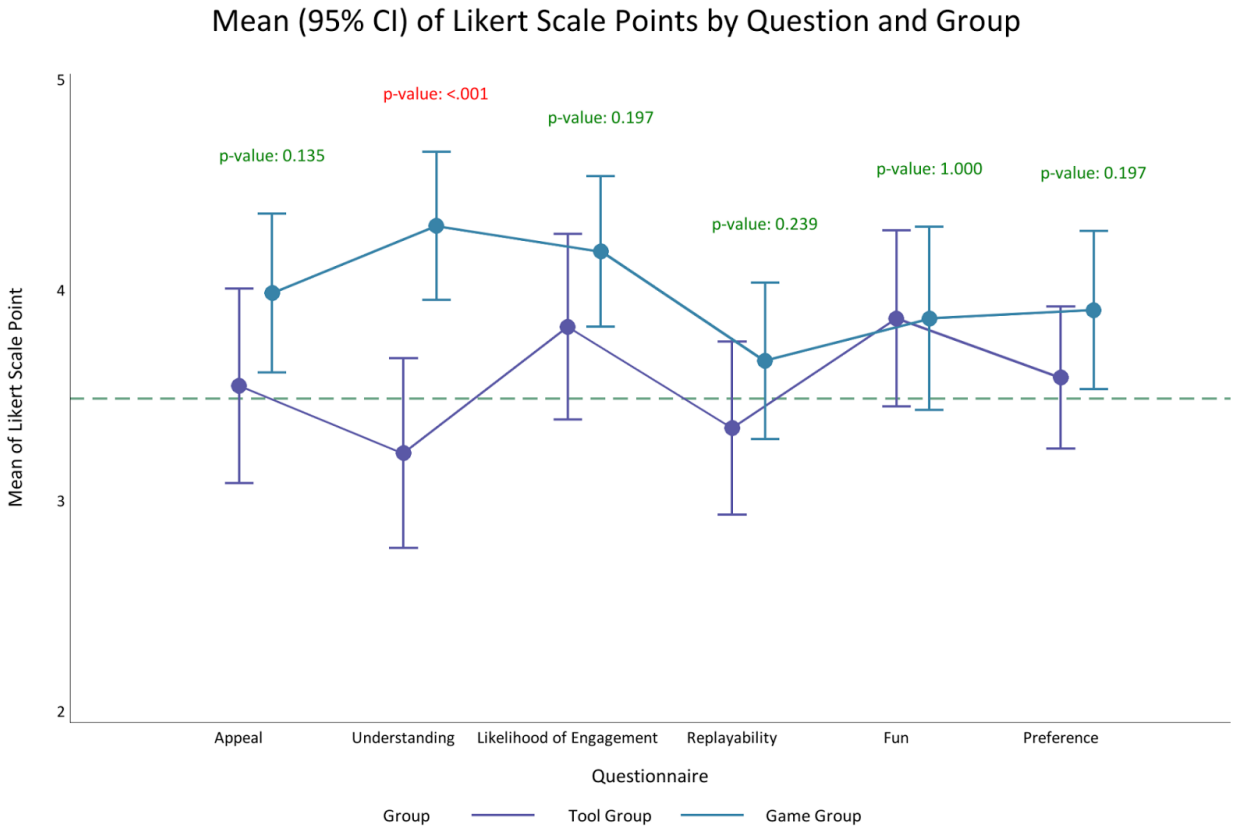

Figure 21: Mean of Likert Scale Points ordered by Questions and Group, with 95% confidence interval.

4.3.3. Open-Ended Questions

We looked into the open-ended questions and tally the feedback, looking for any trends that may appear during the evaluation process. Labels are marked next to the result, so as to indicate what graph the text is referring to. The first category is the Appeal, which measures how likely the software appeals to the participants. The results are shown on the next page. (Figure 23)

As you can see, there is no significant difference found in both groups. We therefore say both groups have very similar game appeal towards both versions of the software, therefore, no obvious trends can be found.

The next category is the Features, which measures which particular aspects, such as game features, mechanics, user experience, or anything that stuck out with the impression, is effective or ineffective. The results are also given on the next page. (Figure 24)

Since the editor is not present in the Game group's testing session, any feedback gathered are about the gameplay. What we find was 9 out of 15 participants in the Game group are in favor of the gameplay, while 8 out of 11 participants in the Tool group responded the editor feature is effective. No participants in the Tool group said the editor is ineffective. However, 10 participants in total from both groups responded "No", which may be interpreted as "the software is not that impressive enough to say it is effective or ineffective."

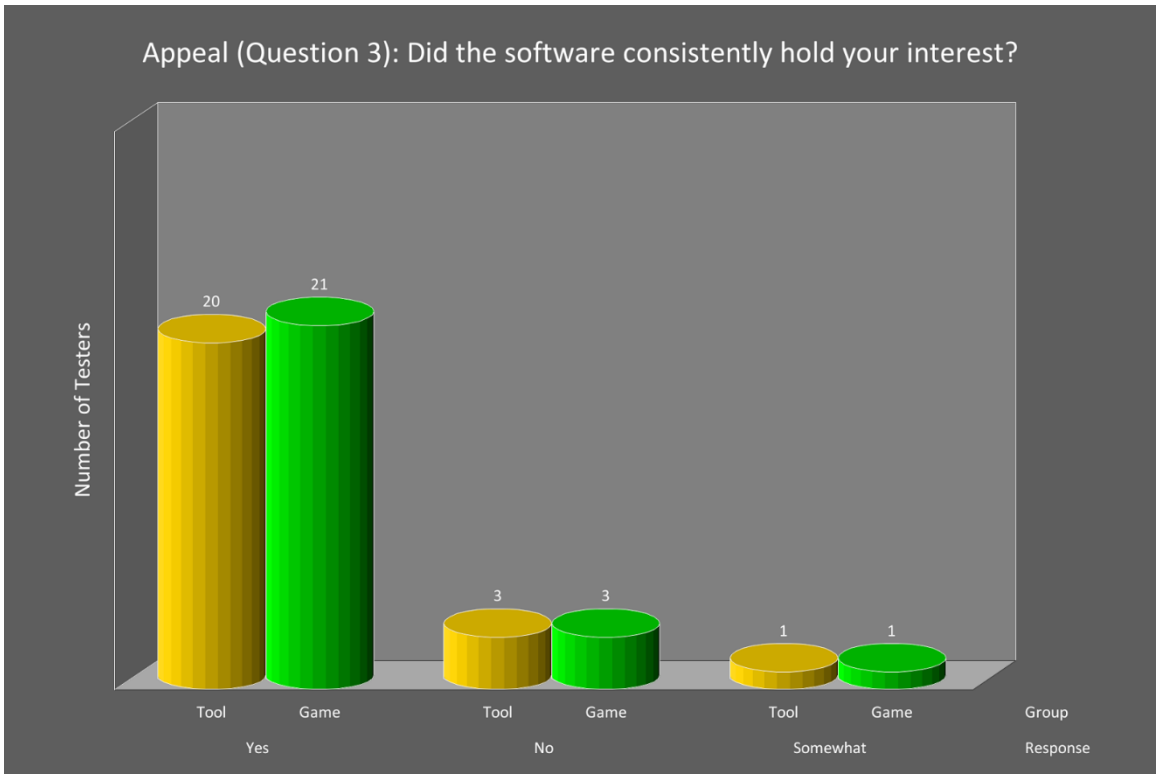

Figure 22: Graph of the number of participants answering the open ended question related to Appeal.

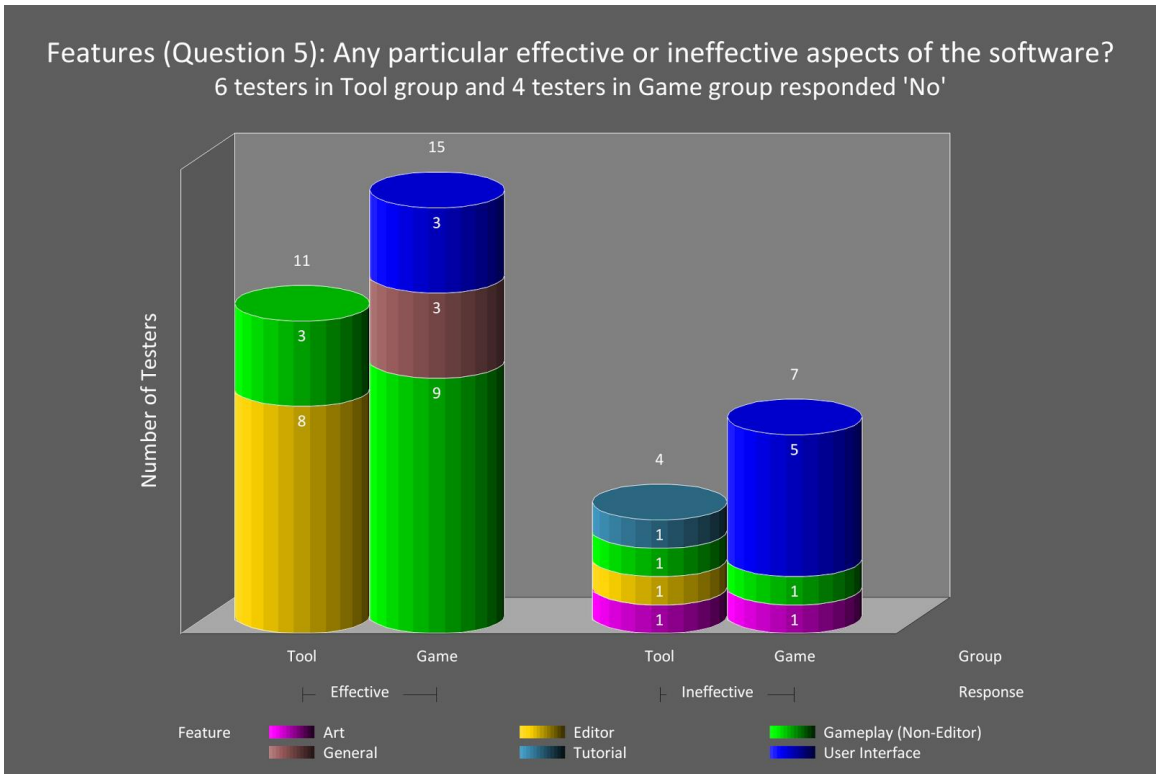

Figure 23: Graph of the number of participants answering questions about effectiveness of software features and other aspects.

For the next Features feedback question, we asked the participants what features from the software to retain, or improve. 28 out of 50 participants from both groups would like to keep the game itself in. 9 participants from the Tool group like to keep the editor in, while 6 participants in the Game group want to keep the User Interface as it is in. See the following graph for the results. (Figure 25)

Both groups have a significant decrease in the number of requests for improvements, compared to number of requests to retain features. The Tool group expressed the editor and the user interface in general needs to be improved, while the Game group expressed the user interface and the gameplay to improve, and 3 participants in the Game group requested to have an editor. Interestingly, the majority of the Tool group does not express interests in improving the gameplay when compared to the Game group, suggesting that having an editor will very slightly improve the gameplay. The results are shown below. (Figure 26)

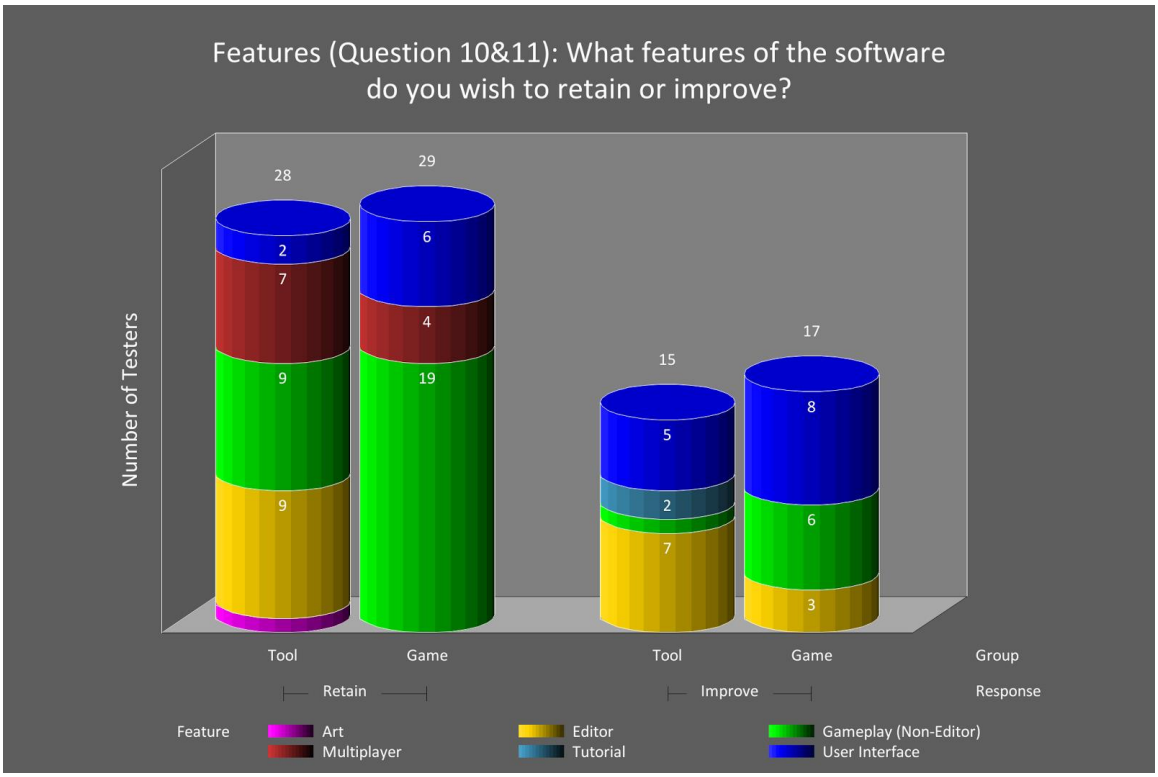

Figure 24: Graph showing the number of participants requesting features to keep and features to improve.

When asked what software features are the best and the worst, the participants from both groups are nearly consistent with the features they wished to retain. Over half of the participants out of 12 from the Game group who answered which features are worst did not like the user interface, which is also consistent with the number of requests to improve it.

These numbers suggested a trend in which features the participants liked should be kept in the software, if the software is to implement improvements. These numbers can be seen in the following graph on the next page.

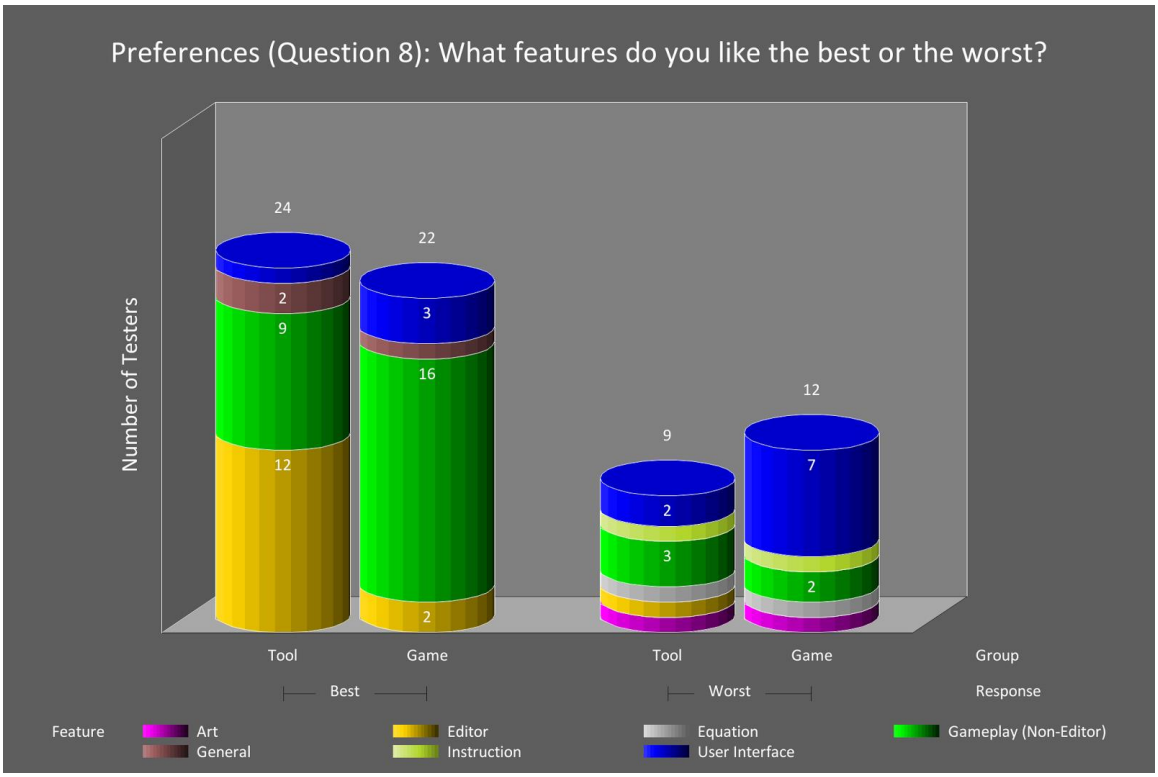

Figure 25: Graph showing the numbers of participants rating the best and worst features from both groups.

When asked about replayability of the software's current state and the software's future state with improvements, a total of 19 participants from both groups said they have the desire to continue playing at its current state, and a total of 6 participants said they will not play even when the software has improved. This suggests that if more improvements are made, the more likely the end users will continue to use the software. The results are shown in the following graph. (Figure 27)

The next feedback question asked is about understanding. We want to know how likely it is for an end user to pick up the software and be able to work with the software. To most participants, the software is easy to pick up and play around with, but 7 out of 25 participants in the Tools group responded the editor feature is confusing, and 3 out of 25 participants in the Game group responded the mathematical equations used are confusing. See the results shown in the following graph on the next page. (Figure 28)

18 participants in the Tool group and 17 participants in the Game group both expressed they would like to understand the rules and objectives prior to testing the software, while 22 participants in the Tool group and 20 participants in the Game group responded the instructions were explained very clearly to them. 45 out of 50 participants in total fully understands what this software is and are able to describe this software to others.

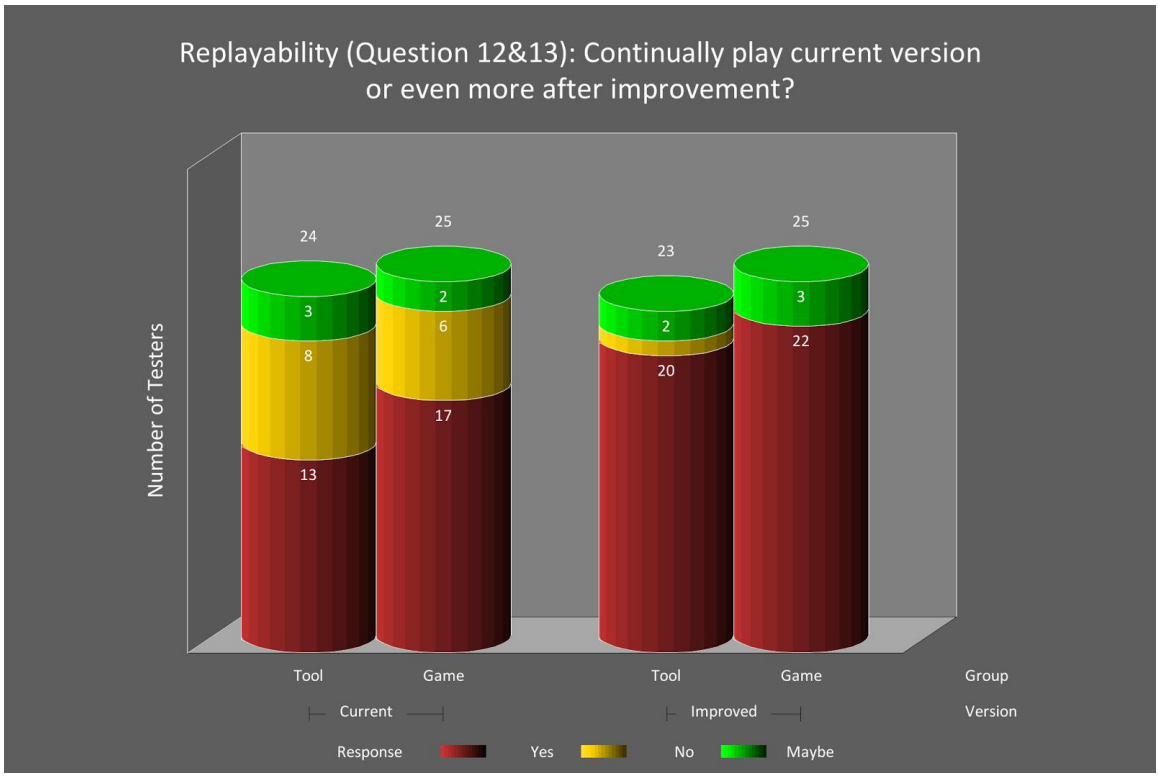

Figure 26: Graph showing the number of participants willing to continue playing the software at its current and improved state.

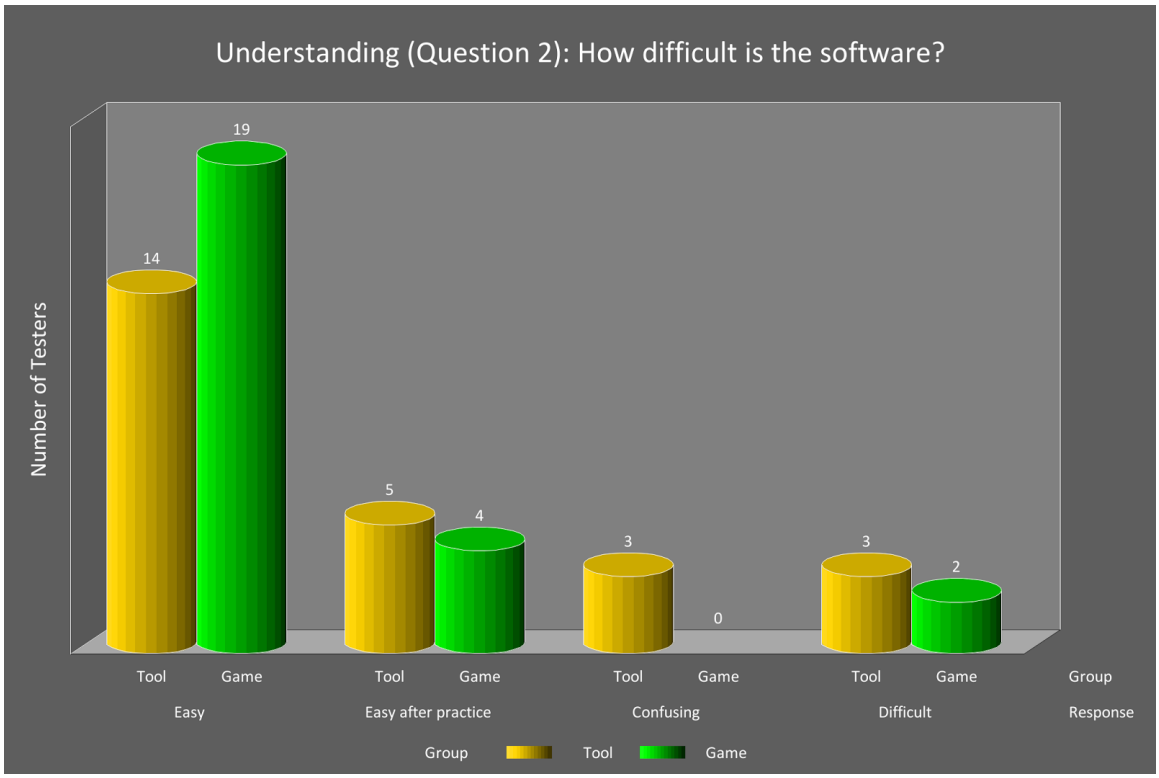

Figure 27: Graph showing the number of participants rating the difficulty of using the software.

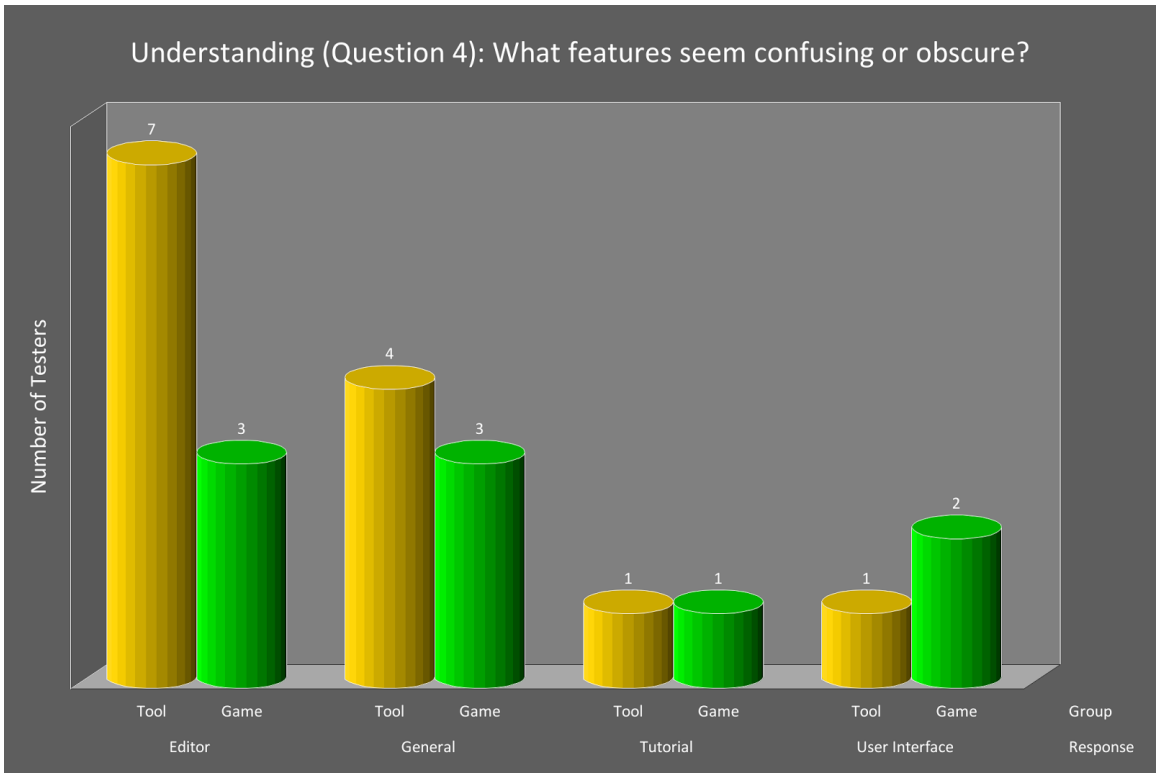

Figure 28: Graph showing how many participants are confused of certain features.

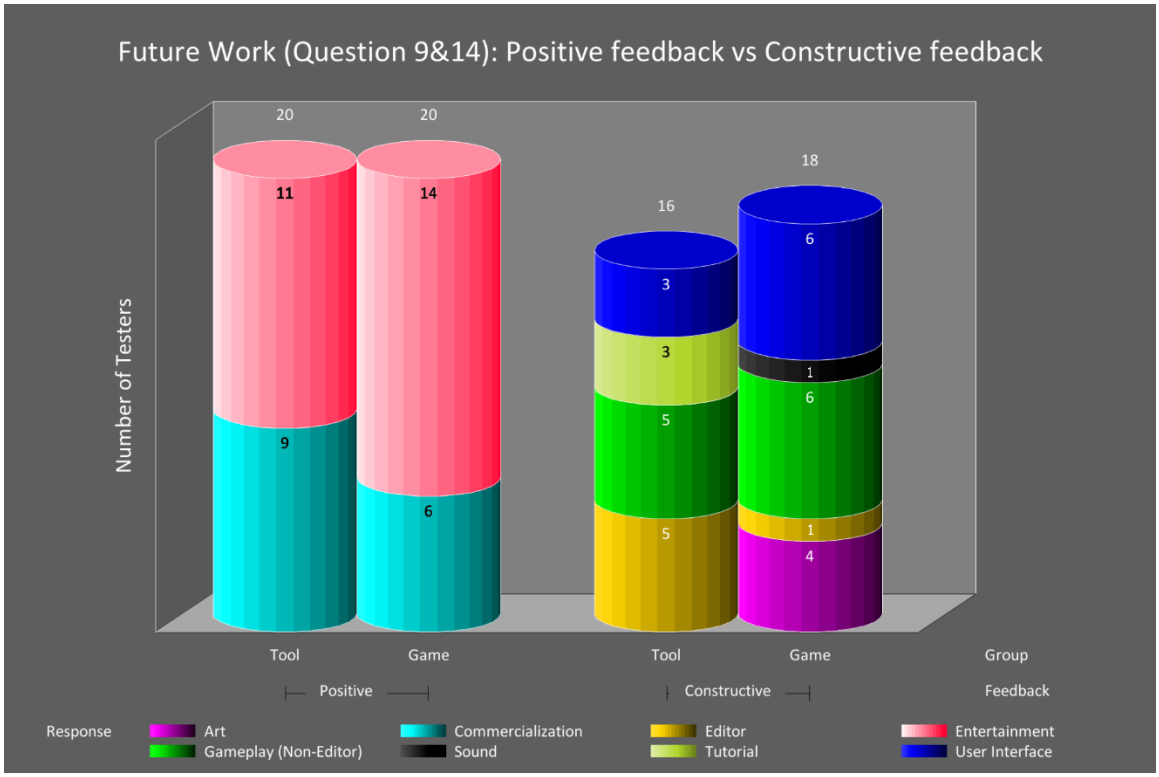

Figure 29: Graph showing player feedback, categorized into many different sections.

In the graph shown (Figure 29), 7 participants out of 25 in the Tool group are confused with the editor, while 3 out of 25 participants in the Game group are confused with the numbers of the presets set for them to choose. The Tool group has slightly more participants being confused about the software than the Game group.

Finally, for the last graph (Figure 30) showing how the feedback are categorized, 25 participants responded the software is fun to play or the game is good, while 15 participants suggested to publish the software and commercialize it. All constructive feedback from Questions 8, 9, 10, 11, 14 (Figure 25, Figure 26, Figure 30) are consistently pointing out that there is still room for improvements on game features, such as the editor and the gameplay.

5. Conclusion

The results between the Likert scale questions and the open-ended questions were inconsistent, since the participants in each group only tested the version they are assigned to. The data was not collected equally, therefore, the evidence that the extra editor features in the software was appealing to the participants is insufficient.

In the Likert scale questions, even though there is a significant difference between the Tool and Game groups, the Game group seems to have a better understanding of this software than the Tool group. Based on the responses in the open-ended question, however, the software is not difficult for them to play.

To explain, from the results obtained from our groups of participants, we conclude that having an extra editor feature added into the game poses no significant differences when compared with the version without it. Our sample size is not significantly large enough to see if one is better than the other. The feedback from both the Likert scale questions and the open-ended questions do not suggest adding a technical game feature to a game brings up the complexity of understanding the game nor does it improves the gameplay more than before the game feature is added in. but it hinted this possibility. We feel if there are more participants involved, and the post-test surveys are done more correctly, we may be able to draw out more clear results and not rely on assumptions based on the hints given.

We assessed there are many factors from both versions of the software that we can identify as criticisms. One big factor, according to the feedback gathered from the open-ended questions in the survey, is the confusing graphical user interface (GUI) shown to the end users. The big difference in both of the software versions used for both the Tool group and the Game group, respectively, is the editor GUI. Since the editor GUI needs to depict unit attributes and changes made while the player is modifying them, they take up a large portion of the screen filled with text so it can show the information to the end users. This abundance of text may have put off and distracted the end users from obtaining the right information.

Another big factor is the gameplay of the software, of which both groups report it as confusing. It could be due to how the game is played, such as splitting and merging or the unit interactions, or the GUI not presenting relevant information correctly to the end users. It is also possible the aesthetics of the game, such as appearance, graphical details, and the lack of visual cues may contribute negatively to the end users in not understanding what the game is all about, or what is going on in the game.

Improvements can be made by finding way to minimally depict the details to the end users, showing only the high level portions of the editor GUI. More can be said in the next section.

6. Future Work

Software development is an ongoing iterative process. With the feedback gathered from the post-test survey, there are many necessary changes needed. Overall improvements include restructuring and re-planning the GUI design can be made to help create a consistent user experience, and to show the right information to the end users.

For the editor, one of many approaches to improve for how it is displayed to the end users is to hide it partially. Only show the editor when the player demands it. This helps to keep the end users focused on the task at hand, and have them digest new information when they ask for it. To improve upon the user interface, it may be a good idea to partially show a bit more information to the end users. This should slightly improve the user experience, and does not affect too much that the end users are overwhelmed by the amount of information.

For the gameplay rules, according to the feedback from the participants, an interactive tutorial mode can be considered to teach the end users how to play. However, an interactive tutorial mode may not be suitable for games with mechanics that can be discovered through experimentation, such as ours.[50] Graphical improvements are useful in giving the end users visual information and feedback cues. There can be changes added to the game, to indicate the interactions of the game objects, or make certain areas of the software to be more obvious to the players.

7. Postmortem

This section explains the woes and troubles the author has to go through, and shares his experiences to interested readers. This section is written in first-person perspective and in the style of a blogpost, since it is about the author of this project reflecting upon the research that has been done, and what we all can learn from.

7.1. What went right?

7.1.1. Choosing Unity

Before the end of summer, I had done my research on what game engine supports multiplayer natively. This means it is the responsibility of the engine developers to maintain network multiplayer support, as well as provide an easier way for developers using the game engine to use the network system. At the time, Unity and Unreal Engine had just been released for free for personal uses, but only Unity supported network multiplayer right from the beginning.

[50] (Andersen, et al., 2012)

Knowing almost little to nothing in C# and Unity in general, I decided to embark on the journey of learning how to use Unity. Thinking back on this decision, I feel I have made the right choice in learning how to use a new game engine. It helps to have another set of skillset, in my opinion.

7.1.2. Scoping the Project

I knew from the beginning that I only had less than 1 year to complete this Master's project. This means I need to think of a project that will take about 1/3 of the year to complete, thanks to taking a course on Production Management for Interactive Media.

As it turns out, the entire scope of the project lasted exactly just 11 months, which is an awesome feat of accomplishment on predicting how long the project development will take. I would highly recommend readers to scope their projects down to 1/3 of the original total planned time. It really helps once you are close to the end of the development phase.

7.2. What went wrong?

7.2.1. Tutorial Manager

I've learned my lessons when it comes to creating a tutorial for my game. So I want others to not follow my footsteps when they are also working on, or are about to start working on the tutorials.

What not to do #1 - Make 1 Tutorial Manager managing every single item, when they all can be broken down into modular components.

What boils down to is, my game is never designed from bottom-up to be very modular. I haven't used Open-Closed Principle when I was making my game, so the whole game was built with "rewrite every scene from scratch for every additional modes". This also means, the tutorials cannot be made to allow players to issue commands or do their own thing, while the tutorial "guides" the players around. It's just not possible, unless I redid everything from scratch.

Instead of doing a rewrite due to time constraints, I had to resort to static animations and force the players to read monotonous dialogues. That would not give a good game experience overall for the players in the long run.

What not to do #2 - Write all dialogues inside your Tutorial Manager script.

This is especially true if you want to have a tutorial that will be modified over and over again until it is right.

For me, since I had to create the game in Unity Web Player as a browser game, I didn't think much on what to do with this, and decided that I will be writing the dialogues up in a C# class object and make it easier to set and get. I was actually lucky that I do not have to work a lot on the tutorial dialogues. But thinking how hard it is to modify the dialogues, I think it would be best to put it in this short list.

What not to do #3 - Completely separate animations, scripted dialogues, and scripted events.

This is key. Like what I mentioned before, I have to do everything from scratch when I'm adding additional game modes to the game, and that includes the tutorial mode. Since all game objects are not built with modularity in mind, there is not much I can do except to "wing" it, and pretend that I have something to show to them.

This means, I would have to find some way to manage tutorial animations, scripted dialogues, and other events that needs to occur for the players to understand what's going on. I managed to separate all three of them, and they work nicely, except for a few major flaws.

- You cannot rewind the tutorials.
- You need to restart the tutorials from the very beginning in order to get to a certain point the players missed out on.
- You have to exit the tutorials first.
- You cannot easily move anything around. If you have a script that needs to be moved earlier, everything needs to be rewritten.
- It is hard to get the timings right.
- You cannot change the length, the width, and the height of dialogues.
- It is definitely hard to track down weird bugs that would work normally in some cases, but not other cases.
- It is painful to fix when you are dealing with free aspect ratios. Good thing you can give fixed aspect ratios on some websites.
- You are limited to publishing on those websites.
- You are limited in any other ways.
- You are limited to a certain Unity game engine (because I'm using Unity Web Player).

Yes, the last one is really harsh on my development. Because every significant component of a good tutorial is effectively affected poorly.

With just the tutorials alone, I have my internal testers complaining very much on the flaws of the tutorials to the point they do not want to read a few **paragraphs**. Yes, I said paragraphs, because I cannot modify the dialogues. I can do something with the UI, but the dialogues are not affected by that in any way.

Still, this last bit is crucial to me. I do not have any ideas on how to create **interactive** scripted tutorials. So I had no other options but to make the tutorials I am using right now. In the future, I'll first do a bit of research on how to make interactive scripted tutorials when the time comes for me to make a tutorial, then try to build a foundation for making game objects more modular for use with tutorials and the game.

7.2.2. Multiplayer and "New Multiplayer"

There was a point where I have just finished Multiplayer mode for the project. It was working fine, but bugs are much more frequently appearing, and it is really hard to get the client have the same variable values from the server sometimes. It was getting annoying to the point that I decided to redo Multiplayer mode from scratch, 3 weeks late than I thought I would start doing.

In hindsight, when I noticed there is trouble with syncing variable values, I knew that was the moment where I need to rewrite the codebase. But the pressing feeling of "Oh, I can go ahead and fix this. No worries," is actually the driving force for delaying the rewrite. It is the train of thought that I have yet to overcome when fixing a feature that was already complete and working.

In the end, the rewrite was completed, and now I am much happier with how it turned out. Variables sync more reliably from the server to client and vice versa, and the issues of desynchronizations have decreases a lot. It is still there as of writing, but it is much easier than what was before.

Always rewrite from scratch if you think something underlying the mess of code is wrong.

7.2.3. Post-Test Survey Could Have Been Better

The test procedures conducted on the participants are all successful. They were all briefly introduced to the software, and were explained on what the software does as they are using it. Once done with the software, they were then handed out the post-test survey. And this is where it begins to fail.

When doing an objective comparison, the testers need to know what things are selected for comparison, and they need to know they are testing those things out beforehand. Once the testers did their playtesting sessions, they can then decide which is better, and write their opinions down. It is this part where I did my post-test survey incorrectly, meaning the testers themselves do not know what the differences are for the two software builds, and therefore cannot point out which software has a higher appeal.

I could have done a different test from the start, where Software A is tested first for 1 half of all testers, and Software B is tested first for the other half, so as not to introduce any bias, particularly when the first option given to the participants is more likely to be chosen as more favorable[51], regardless if the first option is Software A or Software B.

I could also have done the post-test survey differently by not asking open-ended questions and asking multiple-answers questions instead, so it makes quantifying the results easier for me to do. Instead of feedback and suggestions for software improvements, those same types of feedback can be converted into a table with numbers that readers can understand.

[51] (Cerney & Banaji, 2012). The study reveals when given two options, participants are more likely to choose the first option than the second option.

7.2.4. Being a Sole Developer

The biggest advantage of being a sole developer on this project is I have the biggest freedom in deciding what I want to do with the project. The biggest drawback, however, is the quality assurance. I do not always have the ability to spot errors while developing the software, nor do I get the chance to find a good software tester in the area of where I live.

As of writing this paper, there are still hidden bugs and broken code in the software, but I believe it is due to Unity deprecating support for Unity Web Player, which I used for this project. I could not devote my free time testing and debugging the software just to find the bugs out, or try to migrate to a different platform. What I could have done is have a stricter procedure in assessing software unit test cases or use some methodology to help me debug. But alas, I never did all those things. I could not even find encouragement and motivation to do those things.

If you want to get encouragement or motivation to continue with the project, the best way is to find a partner that works with you. There are many benefits of having a partner, and if you are just starting out on working on a project, I really, really suggest you to not work alone. I would also suggest you not to find anyone who will not devote their time to work with you together on a project.

7.3 What did I learn?

7.3.1. Things do not always go the way you want them to go

During the course of development in Unity, there are multiple times where the Unity would not cooperate with me. For example, having multiple [SyncVar] variables placed onto a game object, and Unity will sometimes set values incorrectly, even though they are supposed to sync the variable values across the network. When this happens, I would have trouble tracking down why it is not working the way I wanted them to be. It takes a lot of time debugging and asking questions on the forums, and you will sometimes get answers every other day.

Or how about the time when the function does work, but it turns out there was an upper limit on the number of usages you are allowed to make. Many developers complained of this, and subsequently the Unity3D developers had to make some compromises to get the functionalities to work. The wait for the updates lingers longer than anticipated sometimes.

Game engines developed by other organizations, such as Unity3D, means those game engines are hard for other developers who use them to fix when they come across issues with the engines.[52] And they would choose to make their own engine so they can have total control over every aspect for their games. I did not go through this, because I know the feature that I wanted to use, which is the network multiplayer capabilities, is really tough for a sole developer to work on, let alone completing the entire project as a whole. It is just not possible, and for that, I learned that it is a good compromise for me to use an existing game engine than to develop a game engine from scratch, and facing the troubles of the engines not working with you is a hurdle this compromise brings along. Accepting this fact is easier, I feel.

[52] (StackExchange, 2016)

7.3.2. Never Use Planned Deprecated Software

Before the project had started, I was very aware Unity will be dropping support for Unity Web Player, but no official announcements have been made warning the actual date it will stop supporting it. I trudged along, hoping that by the time I finish the project, I will no longer need to touch it again. I was almost right and wrong.

I am right about the fact that I will not be touching the project the moment I complete this project. I was wrong about the same fact, because I need to have the project working correctly before I complete all of this. To be honest, I am indifferent about this, because I do not feel happy nor sad when reaching the end.

But if you have to use a deprecated software that is planned to be deprecated, use the tools available and work only with the tools provided. For me, I have Unity Editor running the software just fine, but I have trouble working with Unity Web Player, which is planned for deprecation. Instead of relying on such software, I prefer myself to use the tools available, which is the Unity Editor. That helped me get through really tough situations, but it would be nice if I only resort to using the Editor at the worst possible scenario.

8. References

Adams, D. (2006, September 11). *Company of Heroes Review*. Retrieved from IGN: http://www.ign.com/articles/2006/09/11/company-of-heroes-review-2

Adams, E. (1998, October 16). *Designer's Notebook: A Symmetry Lesson*. Retrieved from Gamasutra: http://www.gamasutra.com/view/feature/131699/designers_notebook_a_symmetry_.php

Andersen, E., O'Rourke, E., Liu, Y.-E., Snider, R., Lowdermilk, J., Truong, D., . . . Popovic, Z. (2012). The Impact of Tutorials on Games of Varying Complexity. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 59-68). New York, NY, USA: ACM. doi:10.1145/2207676.2207687

Bangay, S., & Makin, O. (2013, September 23-25). Modelling Attribute Dependencies in Single Unit. *Games Innovation Conference (IGIC), 2013 IEEE International*, 20-26.

Bergensten, J. (2008, November 26). *RTS Game-play Part 5: Introduction to Unit Balancing*. Retrieved from Oxeye Game Studio News & Development Blog: http://www.oxeyegames.com/rts-game-play-part-5-introduction-to-unit-balancing/

Blackbird Interactive. (2016, March 26). *Homeworld: Deserts of Kharak*. Retrieved from Homeworld: Deserts of Kharak: http://www.desertsofkharak.com/

Blizzard Entertainment. (2002, July 3). *Warcraft 3: The Reign of Chaos*. Retrieved from Blizzard Entertainment: http://us.blizzard.com/en-us/games/war3/

Blizzard Entertainment. (2009, March 24). *Rookie Mistakes*. Retrieved from Battle.net: https://web.archive.org/web/20090324034745/http://classic.battle.net/war3/basics/rookiemistakes.shtml

Blizzard Entertainment. (2015). *StarCraft II*. Retrieved from Blizzard Entertainment: http://us.battle.net/sc2/en/

Blizzard Entertainment. (2016, April 27). *Carrier*. Retrieved from Battle.net: http://us.battle.net/sc2/en/game/unit/carrier

Breslin, S. (2009, July 16). *The History and Theory of Sandbox Gameplay*. Retrieved from Gamasutra: http://web.archive.org/web/20160419212023/http://www.gamasutra.com/view/feature/132470/the_history_and_theory_of_sandbox_.php?print=1

Burgun, K. (2011, June 8). *Understanding Balance in Video Games*. Retrieved from Gamasutra: http://web.archive.org/web/20140512114845/http://www.gamasutra.com/view/feature/134768/understanding_balance_in_video_.php?print=1

Burgun, K. (2012, March 29). *What Makes a Game?* Retrieved from Gamasutra: http://web.archive.org/web/20160126142357/http://www.gamasutra.com/view/feature/167418/what_makes_a_game.php

Byte Publications. (1982, December). Table of Contents. *Byte: The Small Systems Journal, 7*(12), p. 5. Retrieved March 20, 2016, from https://archive.org/stream/byte-magazine-1982-12/1982_12_BYTE_07-12_Game_Plan_1982#page/n3/mode/2up

Cerney, D. R., & Banaji, M. R. (2012). First is Best. *PLoS ONE, 7*, 1-5. doi:10.1371/journal.pone.0035088

Cheng, D. C., & Thawonmas, R. (2004, November). Case-based Plan Recognition for Real-Time Strategy Games. *Proceedings of 5th Game-On International Conference on Computer Games* (pp. 36-40). Reading: University of Wolverhampton Press.

Devkar, A. (2003, March 18). *Command and Conquer in the Development of Real-Time Strategy*. Retrieved from http://web.stanford.edu/group/htgg/sts145papers/adevkar_2003_1.pdf

Dijkstra, D. E. (1961, November). ALGOL-60 Translation. *Mathematisch Centrum*. Amsterdam, Netherlands: Stickhting. Retrieved from http://web.archive.org/web/20160312000325/http://www.cs.utexas.edu/~EWD/MCReps/MR35.PDF

Dinosaur Polo Club. (2015, August 28). *Mini Metro - Beta31: Audio!* Retrieved from Steam Community: http://steamcommunity.com/games/287980/announcements/detail/800867231024886989

Dulin, R. (1997, October 1). *Total Annihilation Review*. Retrieved from Gamespot: http://www.gamespot.com/reviews/total-annihilation-review/1900-2535174/

Egenfeldt-Nielsen, S., Smith, J. H., & Tosca, S. P. (2012). *Understanding Video Games: The Essential Introduction* (2nd ed.). New York, NY: Routledge.

Ericsson, K. A., Krampe, R. T., & Tesch-Romer, C. (1993). The Role of Deliberate Practice in the Acquisition of Expert Performance. *Psychological Review, 100*, pp. 363-406.

Fayard, T. (2007). Using a Planner to Balance Real Time Strategy Video Game. *Workshop on Planning in Games , ICAPS, vol. 2005.*, 1-8.

Gallegos, A. (2011, November 23). *Minecraft Review*. Retrieved from IGN: http://www.ign.com/articles/2011/11/24/minecraft-review

Geryk, B. (2001, June 11). *A History of Real-Time Strategy Games*. Retrieved from Gamespot: https://web.archive.org/web/20010611023323/http://gamespot.com/gamespot/features/all/real_time/index.html

Giant Bomb. (2016, March 26). *Macromanagement*. Retrieved from Giant Bomb: http://web.archive.org/web/20160326224102/http://www.giantbomb.com/macromanagement/3015-484/

Goetz, P. (2006, August 23). *Too Many Clicks! Unit-Based Interfaces Considered Harmful*. Retrieved from Gamasutra: http://www.gamasutra.com/view/feature/1839/too_many_clicks_unitbased_.php

Griliopoulos, D. (2008, September 16). *Multiwinia UK Review*. Retrieved from IGN: http://www.ign.com/articles/2008/09/16/multiwinia-uk-review

Hafer, T. (2015, February 12). *Total War: Attila Review*. Retrieved from IGN: http://www.ign.com/articles/2015/02/12/total-war-attila-review

Hale-Evans, R. (2007, June 27). *Abstract Strategy Games*. Retrieved from Seattle Cosmic Wiki: http://web.archive.org/web/20160417063649/http://www.ludism.org/scwiki/AbstractStrategyGame

Hassey, P. (2016, April 23). *Galcon*. Retrieved from Galcon: https://www.galcon.com/

Hastings, E. J., Guha, R. K., Member, L., IEEE, & Stanley, K. O. (2009, December). Automatic Content Generation in the Galactic Arms Race Video Game. *IEEE Trabsactions on Computational Intelligence and AI in Games, Vol. 1, No. 4*, 245-263.

International Abstract Games Organization. (2010, March 7). *Abstract strategy games and other genres out of scope of IAGO*. Retrieved from International Abstract Games Organization: http://web.archive.org/web/20100307130807/http://iagoweb.com/wiki/game-genres

Johnson, D. M. (2013, September 7). *Real-Time Strategy "Level Design"*. Retrieved from Ultima Ratio Regum: http://www.ultimaratioregum.co.uk/game/2013/09/07/real-time-strategy-level-design/

Kleinberg, J. (2011, September 23). *Networks: Course Blogs for INFO 2040/CS 2850/Econ 2040/SOC 2090*. Retrieved from Cornell University: http://blogs.cornell.edu/info2040/2011/09/23/real-time-strategy-and-game-theory/

Lahiri, S. (2010, October 4). *Mind Games of a Tactical Kind*. Retrieved from Slant Magazine: http://www.slantmagazine.com/house/article/mind-games-of-a-tactical-kind-ruse

Lara-Cabrera, R., Cotta, C., & Fern´andez-Leiva, A. J. (2012). Procedural Map Generation for a RTS Game. *13th International GAME-ON Conference on Intelligent Games and Simulation* (pp. 53-58). Malaga, Spain: Eurosis.

Lara-Cabrera, R., Cotta, C., & Fernández-Leiva, A. J. (2013). A Procedural Balanced Map Generator with Self-adaptive Complexity for the Real-Time Strategy Game Planet Wars. *EvoApplications 2013, LNCS 7835*, 274–283.

Lara-Cabrera, R., Cotta, C., & Fernández-Leiva, A. J. (2014). On balance and dynamism in procedural content generation with self-adaptive evolutionary algorithms. *Springer Science+Business Media Dordrecht*, 157–168.

Lara-Cabrera, R., Nogueira-Collazo, M., Cotta, C., & Fernández-Leiva, A. J. (2015). Procedural Content Generation for Real-Time Strategy Games. *International Journal of Artificial Intelligence and Interactive Multimedia, Vol. 3, Nº 2.*, 40-48.

Li Yan, Y. S. (2014). An Interactive Path Planning Method Based on Fuzzy Potential Field in Game Scenarios. *Foundations of Intelligent Systems: Proceedings of the Eighth International Conference on Intelligent Systems and Knowledge Engineering* (pp. 519-529). Shenzhen, China: Springer Berlin Heidelberg.

Looney Labs. (2016, March 14). *Icehouse*. Retrieved April 17, 2016, from Looney Labs: http://web.archive.org/web/20160314030212/http://www.looneylabs.com/rules/icehouse

Mark Hendrikx, S. M. (2013, February). Procedural Content Generation for Games: A Survey. *ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 9, No. 1, Article 1*, 22.

Miniclip. (2016, April 19). *Agar.io*. Retrieved from Agar.io: http://agar.io

Nash, J. F. (1950, January 15). Equilibrium Points in N-Person Games. *Proceedings of the National Academy of Sciences of the United States of America, 36*(1), 48-49. Retrieved March 26, 2013, from http://www.jstor.org/stable/88031

Nutt, C. (2014, August 8). *'What's your favorite game feature that's often overlooked?'*. Retrieved from Gamasutra: https://web.archive.org/web/20140812051908/http://gamasutra.com/view/news/222980/Whats_your_favorite_game_feature_thats_often_overlooked.php

Ontañón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., & Preuss, M. (2013, December 18). A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *5(4)*, 293-311. IEEE.

Onyett, C. (2007, February 16). *Supreme Commander Review*. Retrieved from IGN: http://www.ign.com/articles/2007/02/16/supreme-commander-review-2

Onyett, C. (2010, March 18). *Command & Conquer 4 Review*. Retrieved from IGN: http://www.ign.com/articles/2010/03/18/command-conquer-4-review?page=1

Onyett, C. (2011, March 16). *Total War: Shogun 2 Review*. Retrieved from IGN: http://www.ign.com/articles/2011/03/17/total-war-shogun-2-review

OpenRA. (2016, March 7). *Veterancy*. Retrieved from Github: http://web.archive.org/web/20160418070213/https://github.com/OpenRA/OpenRA/wiki/Veterancy

Parker, J. (2013, May 10). *Auralux Review*. Retrieved from CNET: http://www.cnet.com/products/auralux/

Pearson, C. (2012, December 5). *Natural Selection 2 Review*. Retrieved from PC Gamer: http://www.pcgamer.com/natural-selection-2-review/

Pedersen, R. E. (2009). *Game Design Foundations* (2nd Edition ed.). Jones & Bartlett Learning.

Pennington, H. (2003, February 5). *Free software maintenance: Adding Features*. Retrieved from Havoc Pennington Personal Homepage: http://web.archive.org/web/20030205234223/http://www.ometer.com/features.html

Shepherd III, F. L. (1997, March 1). The Fog of War: Effects of Uncertainty on Airpower Employment. 40. Montgomery, Alabama: Air Command and Staff College.

Slovic, P. (1995, May). The Construction of Preference. *American Psychologist, 50*(5), 364-371.

StackExchange. (2016, April 18). *Why do game developers write their own engines instead of using existing ones?* Retrieved from StackExchange: http://web.archive.org/web/20160418230907/http://gamedev.stackexchange.com/questions/74388/why-do-game-developers-write-their-own-engines-instead-of-using-existing-ones

StrategyWiki. (2014, October 4). *Warcraft III: Reign of Chaos/Footmen Wars*. Retrieved from Wayback Machine: https://web.archive.org/web/20141004065215/http://strategywiki.org/wiki/Warcraft_III:_Reign_of_Chaos/Footmen_Wars

The Numerical Algorithms Group Ltd. (2012). *Random Number Generators.* Retrieved September 17, 2015, from NAG Library Manual, Mark 23 Online Documentation: http://www.nag.co.uk/numeric/fl/nagdoc_fl23/pdf/G05/g05intro.pdf

Thompson, J. M. (2000, July). *Defining the Abstract*. Retrieved from The Games Journal: A Magazine About Boardgames: http://web.archive.org/web/20160414021707/http://www.thegamesjournal.com/articles/DefiningtheAbstract.shtml

Tocci, J. (2012, April 19). *Five Ways Games Appeal to Players*. Retrieved from Gamasutra: http://web.archive.org/web/20140513043406/http://www.gamasutra.com/view/feature/168807/five_ways_games_appeal_to_players.php?print=1

Todd, B. (2001, August 29). *Conquest: Frontier Wars Review*. Retrieved from Gamespot: http://www.gamespot.com/reviews/conquest-frontier-wars-review/1900-2809129/

Unity3D Technologies. (2016, April 27). Retrieved from Unity: https://unity3d.com/

Walker, M. H. (2004, August 18). *Strategy Gaming: Part V -- Real-Time vs. Turn-Based*. Retrieved from Gamespy: http://web.archive.org/web/20040818124742/http://archive.gamespy.com/articles/february02/strategygames05/

War Drum Studios. (2016, April 19). *Auralux*. Retrieved from Auralux: http://www.auraluxgame.com/

Wayward Strategist. (2014, December 18). *Random Thoughts on Resource Management in RTS*. Retrieved from Wayward Strategist: http://waywardstrategist.com/2014/12/18/random-thoughts-on-resource-management-in-rts/

Wikia. (2014, December 22). *Veterancy*. Retrieved from Company of Heroes Wiki: http://web.archive.org/web/20141222182525/http://companyofheroes.wikia.com/wiki/Veterancy

Wikia. (2016, April 18). *Veterancy*. Retrieved from Supreme Commander 2 Wiki: http://web.archive.org/web/20160418065423/http://supcom2.wikia.com/wiki/Veterancy

Xiong, S., & Iida, H. (2014). Attractiveness of Real Time Strategy Games. *2nd International Conference on Systems and Informatics (ICSAI 2014)* (pp. 271-276). Shanghai: IEEE.

YurdleTheTurtle. (2010, June 24). *How Veterancy Works*. Retrieved from GameReplays.org: http://web.archive.org/web/20160417072005/http://www.gamereplays.org/companyofheroes/portals.php?show=page&name=how-veterancy-works

9. Appendices

All sections will be placed with page breaks, meaning there will be many white spaces in between each section. This is done to preserve formatting and for printing purposes, if one chooses to do so.

The first section is the IRB questionnaires that were handed to the testers/volunteers. The second section contains a journal of a list of Unity3D forum threads, which contains technical solutions and questions related to Unity3D. The goal here is to keep all the solutions together, so future readers can look at this list, and go through the forum threads to learn more about Unity Networking HLAPI. The third and final section contains the original thesis/project proposal of this project. Note, this section contains old information and does not apply to the rest of the paper.

9.1. Post-Test Survey

Institutional Review Board Survey Questions - Multiplier

	Disagree				Agree
This software appeals to me.	1	2	3	4	5
The software itself is easy to understand.	1	2	3	4	5
I like to play against someone else.	1	2	3	4	5
I want to continue playing this software.	1	2	3	4	5
This software is fun to use.	1	2	3	4	5
I want to see more of this type of software.	1	2	3	4	5

- What are the rules and objectives of the software? How did you discover them?

_____
_____

- How difficult is the software?

_____
_____

- Did the software consistently hold your interest?

_____

- Did anything about the software seem confusing or obscure?

_____

- Did any aspects of the software stand out as particularly effective? How particularly ineffective?

_____
_____

- What would it have been good to know about the software before you started playing?

_____

- How would you describe the software to a friend who has never used it?

_____

- What features of the software do you like the best? And the worst?

_____
_____

- What potential aspects do you see about the software? List as many as you can.

_____
_____

- What features of the software do you wish to keep? List as many as you can.

_____
_____
_____

- What features of the software do you wish not to keep? List as many as you can.

_____
_____
_____

- Do you wished to continue using the software as it is right now?

_____

- Do you wished to continue using the software once it is improved even more?

_____

- As a lucky beta tester, do you have any additional comments or questions that you would like to share?

_____
_____
_____

9.2. Project Journal

Making an RTS style game work
http://forum.unity3d.com/threads/making-an-rts-style-game-work.343702/

Unity 5 UNET Multiplayer Tutorials - Making a basic survival co-op
http://forum.unity3d.com/threads/unity-5-unet-multiplayer-tutorials-making-a-basic-survival-co-op.325692/

How to spawn multiple prefabs at the same time through Network Manager?
http://forum.unity3d.com/threads/how-to-spawn-multiple-prefabs-at-the-same-time-through-network-manager.345654/

Consuming inputs: Have a few game objects handle inputs and make others ignore inputs during 1 frame
http://forum.unity3d.com/threads/consuming-inputs-have-a-few-game-objects-handle-inputs-and-make-others-ignore-inputs-during-1-frame.348189/

How to rotate a game object pivoting by the center, and not by the corner?
http://forum.unity3d.com/threads/how-to-rotate-a-game-object-pivoting-by-the-center-and-not-by-the-corner.347670/

Any tips of making an interactive tutorial for games?
http://forum.unity3d.com/threads/any-tips-of-making-an-interactive-tutorial-for-games.347153/

[Solved] How to move dialogs around for interactive tutorial?
http://forum.unity3d.com/threads/solved-how-to-move-dialogs-around-for-interactive-tutorial.347357/

PSA: If you add Network Identity to your game object, your game object will be inactive upon start
http://forum.unity3d.com/threads/psa-if-you-add-network-identity-to-your-game-object-your-game-object-will-be-inactive-upon-start.346562/

How to obtain a Vector3 at this position marked at the red X? (Infographic)
http://forum.unity3d.com/threads/how-to-obtain-a-vector3-at-this-position-marked-at-the-red-x-infographic.346549/

Spawning a game object prefab gives a black color material by default, instead of the material given
http://forum.unity3d.com/threads/spawning-a-game-object-prefab-gives-a-black-color-material-by-default-instead-of-the-material-given.346421/

[Solved] NetworkBehaviour.isLocalPlayer is return false for both the server and the client.
http://forum.unity3d.com/threads/solved-networkbehaviour-islocalplayer-is-return-false-for-both-the-server-and-the-client.345684/

OnGUI() confusion. Need help clarifying for me.
http://forum.unity3d.com/threads/ongui-confusion-need-help-clarifying-for-me.345059/

Why will my server not execute a command sent by the client in Unity 5.1?
http://gamedev.stackexchange.com/questions/102526/why-will-my-server-not-execute-a-command-sent-by-the-client-in-unity-5-1

Client's list of game object prefabs not shown in the Server. How do I solve this?
http://forum.unity3d.com/threads/solved-clients-list-of-game-object-prefabs-not-shown-in-the-server-how-do-i-solve-this.352420/

Looking for a UNET 5.2 video tutorial on spawning objects with client authority.
http://forum.unity3d.com/threads/looking-for-a-unet-5-2-video-tutorial-on-spawning-objects-with-client-authority.356252/

Couldn't understand why the parameter of a [ClientRpc] method is null.
http://forum.unity3d.com/threads/couldnt-understand-why-the-parameter-of-a-clientrpc-method-is-null.356485/#post-2305901

[HLAPI] Chaos and abysmal docs for Lobby, NetworkClient.Ready(), ServerChangeScene
http://forum.unity3d.com/threads/hlapi-chaos-and-abysmal-docs-for-lobby-networkclient-ready-serverchangescene.341565/#post-2309556

[HLAPI] Chaos and abysmal docs for Lobby, NetworkClient.Ready(), ServerChangeScene
http://forum.unity3d.com/threads/hlapi-chaos-and-abysmal-docs-for-lobby-networkclient-ready-serverchangescene.341565/#post-2318215

Add and remove components by scripting
http://forum.unity3d.com/threads/add-and-remove-components-by-scripting.33039/

How to add child GameObjects to parent in Script?
http://answers.unity3d.com/questions/478051/how-to-add-child-gameobjects-to-parent-in-script.html

Execution Order of Event Functions
http://docs.unity3d.com/Manual/ExecutionOrder.html

NavMesh Agent pause?
http://answers.unity3d.com/questions/351049/navmesh-agent-pause.html

Unet errors overriding
http://forum.unity3d.com/threads/unet-errors-overriding.361270/

is it possible to SyncList Objects instead of SyncListStruct?
http://forum.unity3d.com/threads/is-it-possible-to-synclist-objects-instead-of-syncliststruct.360959/#post-2341284

Command pattern in Unet
http://forum.unity3d.com/threads/command-pattern-in-unet.360518/#post-2338433

How to use [SyncVar] on NetworkBehaviour subclasses?
http://forum.unity3d.com/threads/how-to-use-syncvar-on-networkbehaviour-subclasses.360818/#post-2338159

How to parse math expressions in C#? [duplicate]
http://stackoverflow.com/questions/2853907/how-to-parse-math-expressions-in-c

Shunting-yard algorithm
https://en.wikipedia.org/wiki/Shunting-yard_algorithm

Conversion of System.Array to List
http://stackoverflow.com/questions/1603170/conversion-of-system-array-to-list

Scaling the GUI with different screen resolutions
http://answers.unity3d.com/questions/156619/scaling-the-gui-with-different-screen-resolutions.html

[Solved] How to stop game objects from spazzing out when initiating a transform Lerp?
http://forum.unity3d.com/threads/solved-how-to-stop-game-objects-from-spazzing-out-when-initiating-a-transform-lerp.361742/

[Solved] How to show where the player's main camera is located in a minimap?
http://forum.unity3d.com/threads/solved-how-to-show-where-the-players-main-camera-is-located-in-a-minimap.366592/

Hastebin Shader Code made by SigilObscura on freenode, #unity3d
http://hastebin.com/zujaqizara.avrasm

[Solved] How to make team colors?
http://forum.unity3d.com/threads/solved-how-to-make-team-colors.367138/

[Solved] Transparency is not working in this Fog of War shader (pics included)
http://forum.unity3d.com/threads/solved-transparency-is-not-working-in-this-fog-of-war-shader-pics-included.367957/

Create a Fog Shader
http://in2gpu.com/2014/07/22/create-fog-shader/

[Solved] How to raycast from minimap to ground in main camera, without main camera raycasting?
http://forum.unity3d.com/threads/solved-how-to-raycast-from-minimap-to-ground-in-main-camera-without-main-camera-raycasting.368459/#post-2387741

Alternative to OnTriggerStay - to check if an object is inside a collider
http://forum.unity3d.com/threads/alternative-to-ontriggerstay-to-check-if-an-object-is-inside-a-collider.310494/

How would you reduce lag in RTS game?
http://forum.unity3d.com/threads/how-would-you-reduce-lag-in-rts-game.369243/

[Solved] UI Panels are not transformed correctly when Canvas set to "Screen Space - Camera"
http://forum.unity3d.com/threads/solved-ui-panels-are-not-transformed-correctly-when-canvas-set-to-screen-space-camera.369235/

UnityEvent, where have you been all my life?
http://forum.unity3d.com/threads/unityevent-where-have-you-been-all-my-life.321103/

Rotate a Skybox?
http://forum.unity3d.com/threads/rotate-a-skybox.130639/

Setting top and bottom on a RectTransform
http://forum.unity3d.com/threads/setting-top-and-bottom-on-a-recttransform.265415/

Dropdown inside a Scrollview
http://forum.unity3d.com/threads/dropdown-inside-a-scrollview.366232/

[Solved] How to manually spawn a game object onto Network Start Position via script?
http://forum.unity3d.com/threads/solved-how-to-manually-spawn-a-game-object-onto-network-start-position-via-script.371383/

Very useful RectTransform extension methods
http://forum.unity3d.com/threads/very-useful-recttransform-extension-methods.285483/

How to control which direction the Dropdown shows the selections?
http://forum.unity3d.com/threads/how-to-control-which-direction-the-dropdown-shows-the-selections.371162/

Get all children gameObjects
http://answers.unity3d.com/questions/594210/get-all-children-gameobjects.html

How are ratios calculated in an ever-increasing group of units in RTS AI player build orders?
https://www.reddit.com/r/gamedev/comments/3vrew0/how_are_ratios_calculated_in_an_everincreasing/

SyncList not initialized?
http://answers.unity3d.com/questions/1069079/synclist-not-initialized.html

SyncList behaviour has been changed in 5.3.2p4
http://forum.unity3d.com/threads/synclist-behaviour-has-been-changed-in-5-3-2p4.387914/

Resetting network scene objects
http://forum.unity3d.com/threads/resetting-network-scene-objects.369902/

Saving Data Between Scenes in Unity
http://www.sitepoint.com/saving-data-between-scenes-in-unity/

9.3. Original Master's Project Proposal (OLD)

Creating A Real-Time Strategy Game Using Simple Mathematical Equation

A Master's Degree Project Proposal

Submitted by:

Thompson Lee

On September 10th, 2015

To the IMGD Steering Committee

Project Committee:

Project Supervisor: Dean O'Donnell

Project Reader: Brian Moriarty

Project Reader: Charles Rich

Project Thumbnail

This project is to be built as a real time strategy game, where players use simple and easy-to-grasp math equations for generating units and unit attributes. This project aims to pave the way for future procedurally generated real time strategy game unit balancing.

Problem Statement

1. Research Question

A typical real time strategy game requires a lot of gameplay testing to see if units made are balanced for players to play with. Making the balancing process more streamlined for simulating real time strategy units can be done by running algorithms to determine the most optimal unit attributes given. This process allows access to more unit diversity and game designs in the real time strategy genre.

This project needs investigate if this is a viable solution to generating and balancing unit attributes. The best way to do this is by building a real time strategy game using simple mathematical equations to determine usable unit's attributes, and assess the outcome.

2. Previous Work

Procedural content generation in real time strategy games is one of the most interesting challenges in the video game development process.[53] The dynamics in real time strategy games alone vary greatly, especially when involving multiple players of varying skill levels and backgrounds. These dynamics can be treated differently depending on how the contents are procedurally generated, therefore paving the way for many possible research routes.

There are video games that have done research in procedural content generation, in which some of them were able to use procedural content generation methods and techniques, but the usage is somewhat limited to a particular type of game content.[54] It has also been proven that it is possible to have automated content generation in mainstream games.[55] Notable examples include *Minecraft*[56] and *Mini Metro*[57], in which the former uses procedural content generation to generate terrain, and the latter uses procedural audio generation.

There has been research done on production capability for different species of units in a game. Units that rely only on damage per second is not the best, but rather a mix of other unit attributes, such as hit points, defense points, along with other properties, is suggested.[58] Other research involves using procedural map generations built to fulfill requirements in order to maintain interesting and appealing games, suggesting that game balancing can be perfectly

[53] (Lara-Cabrera, Nogueira-Collazo, Cotta, & Fernández-Leiva, 2015)
[54] (Mark Hendrikx, 2013)
[55] (Hastings, Guha, Member, IEEE, & Stanley, 2009)
[56] (Gallegos, 2011)
[57] (Dinosaur Polo Club, 2015)
[58] (Fayard, 2007)

achieved only on extremely dull games.[59] It also theorizes having moderate dynamics and moderate balancing can give ample stimuli to players to expand and to seek their enemies.

Games that have moderate dynamics and balancing can be used as references. *Total War: Shogun 2*[60], *Total War: Attila*[61], and *Multiwinia*[62] are all real time strategy games where unit compositions are similar, and require the players to use strategic unit troop placements on the battlefield to win battles. In these games, the battlefield area is large enough to provide ample stimuli for players to venture out and prepare for battle.

Games with more complicated unit attributes and geographical properties that affect player decisions would be *Starcraft II*[63], *Warcraft III*[64], and *Total Annihilation*[65]. In these games, unit attributes are affected by unit dynamic properties (speed, regeneration, and cooldowns), which are incrementally increased through tech upgrades. It has been shown that unit attributes can determine the outcome of a real time strategy multiplayer game session[66]. Environmental obstacles used in these games, which can lead to players not being able to spot the enemies at a glance can also affect the outcome of the player game session. For example, trees with enemies behind it can block the player's view from seeing the enemies.

Similarly, there are some real time strategy games, such as *Auralux*[67], which utilizes map layouts designed with a blend of *Footmen Wars*[68] in mind. Research has been done exploring map layout and balance in real time strategy games[69], made similarly as *Auralux*.

The game, *Auralux*, provides the basis of linear upgrade paths that players can use during gameplay, as well as taking into account of the map layout. *Footmen Wars* provides the structure of the game, in which each units of different factions have attributes that players can upgrade accordingly, but ultimately, the players can only use that unit for the rest of the game.

3. The Project

The proposed method of evaluating real time strategy gameplay consists of a single unit type which uses a simple mathematical equation to calculate the unit's attributes and properties. It is easier to come up with a sequential relationship than any other complex relationships[70], and is easily deterministic when compared with other units of the same attributes and properties. Structured to be a 2-players only multiplayer network game, it will put emphasis on unit placement and macro-management, will avoid unit power-ups, special abilities, and anything that reduces player's attention, such as micro-management[71].

---

[59] (Lara-Cabrera, Cotta, & Fernández-Leiva, On balance and dynamism in procedural content generation with self-adaptive evolutionary algorithms, 2014)
[60] (Onyett, Total War: Shogun 2 Review, 2011)
[61] (Hafer, 2015)
[62] (Griliopoulos, 2008)
[63] (Blizzard Entertainment, 2015)
[64] (Blizzard Entertainment, 2002)
[65] (Dulin, 1997)
[66] (Bangay & Makin, 2013)
[67] A minimalistic real time strategy game for Android, based in outer space. (Parker, 2013)
[68] A real time tactics custom map game for the real time strategy game, *WarCraft III* and its expansion, *WarCraft III: The Frozen Throne*. (StrategyWiki, 2014)
[69] (Lara-Cabrera, Cotta, & Fernández-Leiva, A Procedural Balanced Map Generator with Self-adaptive Complexity for the Real-Time Strategy Game Planet Wars, 2013)
[70] (Adams E. , 1998)
[71] (Wayward Strategist, 2014)

The project will be only built for human multiplayer with no A.I. bots involved. If the game is built for single player, the game would then be required to provide an A.I. bot for the player to compete against. Creating a new A.I. bot for this purpose is too costly, while getting another human player to play is quicker and more efficient. Therefore, we eliminate single player scenarios from this project, and concentrate on human interactions only. Future work can create A.I. bots for single player gameplay and/or simulation of similar real time strategy games.

There are two viable options of game designs the game project can be built around of. The first option is by considering "massing," in which the players continuously spawn and build the exact same units overwhelming their opponents. It runs the risk of players becoming burdened by the game if not designed correctly[72], and it depends entirely on player skills[73]. The second option is by considering unit group placements, and a bit of micromanagement involved, to determine the outcome of the scenario, used in some games such as *Command and Conquer*[74] and *Supreme Commander*[75]. The players will not have to worry about skills and will put more focus in strategic planning and execution. The main factor to consider the second method is it can be applied to games which use only one type of unit to play, making it easier to evaluate the results.

The game is given a mathematical equation, or an algorithm, that it can use to generate unit attribute values. These attribute values are then set, therefore the units spawned into the game scene will then have the attributes calculated.

$$X_n = X_{n-2} + X_{n-1}, X_0 = 1, X_1 = 1, n \geq 2 \qquad \text{Fig.1}$$

For example, if the Fibonacci sequence algorithm (as shown in Fig.1) is plugged into the game, the resulting unit health attribute values will generate for 10 levels, or tiers. For each level, the unit health will increase up to the number in the sequence that is iterated. Note that it does not have to be a series of numbers from the Fibonacci sequence. It can be any other series of numbers, ranging from randomly generated (pseudorandom number generator, Fig.2[76]), or a sine value ranging from -1 to 1 (sine wave, Fig.3).

$$X_{n+1} = (aX_n + b)\ mod\ m \qquad \text{Fig.2}$$

$$X = sin\theta, 0° \leq \theta \leq 360° \qquad \text{Fig.3}$$

The project will start with a simple mathematical equation that can be used as a multiplier for leveling up units for each tier level. Each tier level can then use the same mathematical equations to level up, until the unit has reached the maximum amount of levels or tiers allowed in the game. In this way, all upgradeable-tiered units will have consistent, linear (or exponentially, depending on the mathematical equations themselves) relationships between two neighboring tech tree nodes, which represents an upgrade that a unit can have in the game.

The mathematical equations mentioned may change to fit within the scope of the production of the project. Since the only concerned component about the project is by using mathematics to determine unit attributes in real time strategy games, it is not to be used as a reliable

[72] (Goetz, 2006)
[73] (Bergensten, 2008)
[74] (Johnson, 2013)
[75] (Onyett, Supreme Commander Review, 2007)
[76] (The Numerical Algorithms Group Ltd., 2012)

method to assess if the game is playable and enjoyable, depending on what mathematical formulae were used. However, it will provide a leeway for future improvements if there is time.

4. Production and Planning

It is to be built using Unity3D 5, a popular game engine with a well-written documentation, a fully active community, and easy-to-obtain online tutorials and instructional videos. Since Unity3D 5 is in constant development, future releases, as well as documentations of new features, may affect the whole project.

The scheduled game development plan for this project is shown on the next page.

Milestones	Deadline	Description
Understand Unity3D	1-Jun-15	Need to familiarize Unity development
Tech Milestones	15-Jun-15	Components for single player clients
Networking	30-Jun-15	Make two clients connect to each other
Unity 5.1 and 5.2	1-Oct-15	New networking features available
Prototype	30-Oct-15	Implement primitive components
Alpha	15-Nov-15	Winning/Losing conditions
Beta	30-Dec-15	Start polishing
Frequent Playtesting	15-Jan-16	Constant iterative development
Assessment	15-Feb-16	Come up with a way to assess experiences
Unscheduled	1-May-16	Future planning in case of emergency

In order to be sure the project is as accurate as possible, it requires constant playtesting, which is worth the effort, as the project can then be tweaked accordingly to provide accurate results over time once the assessments are done.

Evaluation

This project will produce a playable project game, along with results of testing to see if mathematical equations can allow procedurally generated units be used in real time strategy games, regardless of complexity. The results will be a cumulative analysis of overall gameplay experiences of players playing the project game.

Reference

Adams, D. (2006, September 11). *Company of Heroes Review*. Retrieved from IGN: http://www.ign.com/articles/2006/09/11/company-of-heroes-review-2

Adams, E. (1998, October 16). *Designer's Notebook: A Symmetry Lesson*. Retrieved from Gamasutra: http://www.gamasutra.com/view/feature/131699/designers_notebook_a_symmetry_.php

Bangay, S., & Makin, O. (2013, September 23-25). Modelling Attribute Dependencies in Single Unit. *Games Innovation Conference (IGIC), 2013 IEEE International*, 20-26.

Bergensten, J. (2008, November 26). *RTS Game-play Part 5: Introduction to Unit Balancing*. Retrieved from Oxeye Game Studio News & Development Blog: http://www.oxeyegames.com/rts-game-play-part-5-introduction-to-unit-balancing/

Blizzard Entertainment. (2002, July 3). *Warcraft 3: The Reign of Chaos*. Retrieved from Blizzard Entertainment: http://us.blizzard.com/en-us/games/war3/

Blizzard Entertainment. (2015). *StarCraft II*. Retrieved from Blizzard Entertainment: http://us.battle.net/sc2/en/

Dinosaur Polo Club. (2015, August 28). *Mini Metro - Beta31: Audio!* Retrieved from Steam Community: http://steamcommunity.com/games/287980/announcements/detail/800867231024886989

Dulin, R. (1997, October 1). *Total Annihilation Review*. Retrieved from Gamespot: http://www.gamespot.com/reviews/total-annihilation-review/1900-2535174/

Fayard, T. (2007). Using a Planner to Balance Real Time Strategy Video Game. *Workshop on Planning in Games , ICAPS, vol. 2005.*, 1-8.

Gallegos, A. (2011, November 23). *Minecraft Review*. Retrieved from IGN: http://www.ign.com/articles/2011/11/24/minecraft-review

Goetz, P. (2006, August 23). *Too Many Clicks! Unit-Based Interfaces Considered Harmful*. Retrieved from Gamasutra: http://www.gamasutra.com/view/feature/1839/too_many_clicks_unitbased_.php

Griliopoulos, D. (2008, September 16). *Multiwinia UK Review*. Retrieved from IGN: http://www.ign.com/articles/2008/09/16/multiwinia-uk-review

Hafer, T. (2015, February 12). *Total War: Attila Review*. Retrieved from IGN: http://www.ign.com/articles/2015/02/12/total-war-attila-review

Hastings, E. J., Guha, R. K., Member, L., IEEE, & Stanley, K. O. (2009, December). Automatic Content Generation in the Galactic Arms Race Video Game. *IEEE Trabsactions on Computational Intelligence and AI in Games, Vol. 1, No. 4*, 245-263.

Johnson, D. M. (2013, September 7). *Real-Time Strategy "Level Design"*. Retrieved from Ultima Ratio Regum: http://www.ultimaratioregum.co.uk/game/2013/09/07/real-time-strategy-level-design/

Lara-Cabrera, R., Cotta, C., & Fernández-Leiva, A. J. (2013). A Procedural Balanced Map Generator with Self-adaptive Complexity for the Real-Time Strategy Game Planet Wars. *EvoApplications 2013, LNCS 7835*, 274–283.

Lara-Cabrera, R., Cotta, C., & Fernández-Leiva, A. J. (2014). On balance and dynamism in procedural content generation with self-adaptive evolutionary algorithms. *Springer Science+Business Media Dordrecht*, 157–168.

Lara-Cabrera, R., Nogueira-Collazo, M., Cotta, C., & Fernández-Leiva, A. J. (2015). Procedural Content Generation for Real-Time Strategy Games. *International Journal of Artificial Intelligence and Interactive Multimedia, Vol. 3, Nº 2.*, 40-48.

Mark Hendrikx, S. M. (2013, February). Procedural Content Generation for Games: A Survey. *ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 9, No. 1, Article 1*, 22.

Onyett, C. (2007, February 16). *Supreme Commander Review*. Retrieved from IGN: http://www.ign.com/articles/2007/02/16/supreme-commander-review-2

Onyett, C. (2010, March 18). *Command & Conquer 4 Review*. Retrieved from IGN: http://www.ign.com/articles/2010/03/18/command-conquer-4-review?page=1

Onyett, C. (2011, March 16). *Total War: Shogun 2 Review*. Retrieved from IGN: http://www.ign.com/articles/2011/03/17/total-war-shogun-2-review

Parker, J. (2013, May 10). *Auralux Review*. Retrieved from CNET: http://www.cnet.com/products/auralux/

StrategyWiki. (2014, October 4). *Warcraft III: Reign of Chaos/Footmen Wars*. Retrieved from Wayback Machine: https://web.archive.org/web/20141004065215/http://strategywiki.org/wiki/Warcraft_III:_Reign_of_Chaos/Footmen_Wars

The Numerical Algorithms Group Ltd. (2012). *Random Number Generators.* Retrieved September 17, 2015, from NAG Library Manual, Mark 23 Online Documentation: http://www.nag.co.uk/numeric/fl/nagdoc_fl23/pdf/G05/g05intro.pdf

Wayward Strategist. (2014, December 18). *Random Thoughts on Resource Management in RTS*. Retrieved from Wayward Strategist: http://waywardstrategist.com/2014/12/18/random-thoughts-on-resource-management-in-rts/