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 4](#_Toc449153100)

[Abstract 5](#_Toc449153101)

[Acknowledgements 5](#_Toc449153102)

[1. Introduction 6](#_Toc449153103)

[1.1. Terminology of Real-Time Abstract Strategy 6](#_Toc449153104)

[1.2. The Differences: Real-Time Strategy vs. Real-Time Tactics 7](#_Toc449153105)

[1.3. Origins of the Custom-Built Software and How It Ties in to RTAS, RTS, and RTT 9](#_Toc449153106)

[1.4. Game Balance and Methods to Achieve Balance 9](#_Toc449153107)

[1.5. Game Feature and Appeal 16](#_Toc449153108)

[2. Game Design 16](#_Toc449153109)

[2.1. Overview 16](#_Toc449153110)

[2.2. History 17](#_Toc449153111)

[2.3. Game Mechanics 18](#_Toc449153112)

[2.3.1. Attributes Editor 18](#_Toc449153113)

[2.3.2. Splitting and Merging 19](#_Toc449153114)

[2.3.3. Minimap 20](#_Toc449153115)

[2.3.4. Autonomous Attacking 21](#_Toc449153116)

[2.3.5. Lack of Fog-of-War 21](#_Toc449153117)

[2.3.6. A.I. Gameplay 21](#_Toc449153118)

[2.4. How Unit Attributes Relate to Game Design 22](#_Toc449153119)

[3. Resources 22](#_Toc449153120)

[4. Evaluation 23](#_Toc449153121)

[4.1. Research Method 23](#_Toc449153122)

[4.2. Research Question 24](#_Toc449153123)

[4.3. Result 25](#_Toc449153124)

[5. Conclusion 27](#_Toc449153125)

[6. Future Work 28](#_Toc449153126)

[6.1. Software Improvements 28](#_Toc449153127)

[6.2. Procedural Content Generation 28](#_Toc449153128)

[7. Postmortem 29](#_Toc449153129)

[7.1. What went right? 29](#_Toc449153130)

[7.1.1. Choosing Unity 29](#_Toc449153131)

[7.1.2. Scoping the Project 30](#_Toc449153132)

[7.2. What went wrong? 30](#_Toc449153133)

[7.2.1. Tutorial Manager 30](#_Toc449153134)

[7.2.2. Multiplayer and “New Multiplayer” 32](#_Toc449153135)

[7.2.2. Post-Test Survey could have been better 32](#_Toc449153136)

[7.3 What did I learn? 32](#_Toc449153137)

[7.3.1. Things do not always go the way you wanted them to go 32](#_Toc449153138)

[8. References 33](#_Toc449153139)

[9. Appendices 38](#_Toc449153140)

# Table of Figures

[Figure 1: A complete game of Icehouse (Wikipedia) 6](#_Toc449153046)

[Figure 2: Agar.io (left) and Auralux (right). 7](#_Toc449153047)

[Figure 3: An example of three different levels of veterancy, shown by the number of small white bars above their blue health bar. (GameReplay.org) 8](#_Toc449153048)

[Figure 4: Total War: Shogun 2 gameplay. (3D Juegos) 10](#_Toc449153049)

[Figure 5: Total War: Attila gameplay. (Hi-Def Digest) 11](#_Toc449153050)

[Figure 6: Multiwinia gameplay. (NewGamerNation) 11](#_Toc449153051)

[Figure 7: StarCraft II gameplay. (Softpedia) 12](#_Toc449153052)

[Figure 8: WarCraft III: The Frozen Throne gameplay (Blizzard) 13](#_Toc449153053)

[Figure 9: Total Annihilation gameplay. (Wayward Strategist) 14](#_Toc449153054)

[Figure 10: Homeworld: Deserts of Kharak gameplay. (Scientific Gamer) 14](#_Toc449153055)

[Figure 11: Footman Wars gameplay. It is a custom map played in WarCraft III: The Frozen Throne. (Blizzard) 15](#_Toc449153056)

[Figure 12: Galcon gameplay (Phil Hassey). 15](#_Toc449153057)

[Figure 13: Multiplier, the real-time strategy unit balancing tool, a.k.a., the “Software - Tool   
version”. 17](#_Toc449153058)

[Figure 14: The Attributes Editor for Singleplayer Mode, with the Tooltip and the corresponding indication box shown. 19](#_Toc449153059)

[Figure 15: Splitting (left) and Merging (right). 20](#_Toc449153060)

[Figure 16: Minimap and camera boundaries. Note: Text cannot be removed in Unity 5 Education and will always be present. 20](#_Toc449153061)

[Figure 17: The minimap shown without any fog-of-war. Some of the units are taking damage and attacking opponents. 21](#_Toc449153062)

[Figure 18: Graphical interpretation of the results of the 6 5-point Likert scale questions from the post-test survey. 25](#_Toc449153063)

[Figure 19: Mean of Likert Scale Points ordered by Questions and Group, with 95% confidence   
interval. 26](#_Toc449153064)

[Figure 20: Graph of the participants answering the open ended question related to Appeal. 27](#_Toc449153065)

# Abstract

Game features need to demonstrate that they are able to improve things, such as gameplay, game experiences, and such, for a large percentage of players, or make the impossible possible for a smaller user base.[[1]](#footnote-1) We do not exactly know if adding an extra certain game feature can improve a game’s appeal overall. Game developers usually face this problem, and would ask themselves what features should their games include and what not to include. It is especially important when there is a possibility that adding a small game feature can make a big difference to make players notice or love their games.[[2]](#footnote-2)

The purpose of this paper is to evaluate whether adding a particular game feature is attractive and appealing to the players. This game feature, which is an editor built into a custom real-time strategy game, allows the players to customize the attributes of the game units. The evaluation is based on the players’ feedback on the game with or without the editor.

1 out of a total of 51 testers did not complete the survey after testing the game, and is therefore excluded from the evaluation. The rest of the 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.

# Acknowledgements

I would like to thank Dean O’Donnell as supervisor for providing guidance, Brian Moriarty and Charles Rich as readers for providing assistances, 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 Unity community 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 type of video game genre in which the game does not progress incrementally in turns, i.e., all actions are simultaneously executed.[[3]](#footnote-3) 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.[[4]](#footnote-4) Resource management refers to the economics of the available resources the player has access to, and that includes but not limited to: unit resources, ammunition, raw natural resources, territories, and so on.

Abstract strategy games are games where the players think and decide on their moves, and pose minimal or no involvement in luck. Usually, the skill of the players would offset the luck element by a very large percentage of the time.[[5]](#footnote-5) The term mostly applies to board games, card games, and other tabletop games where luck is not the decisive factor. The word, “abstract” refers to games without having any themes, or have themes that are not important to the experience of playing, such as geometric patterns and shapes.[[6]](#footnote-6)

“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*[[7]](#footnote-7) and *Auralux*[[8]](#footnote-8). 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. The term itself, however, is disputed as a non-existing game genre from a game theory perspective[[9]](#footnote-9), and the source of this dispute came from the company, Looney Labs, who used this term to describe one of their games, *Icehouse*.[[10]](#footnote-10)



Figure 1: A complete game of Icehouse (Wikipedia)

This paper will be borrowing the term, “real-time abstract strategy,” to define the custom-built software’s genre mentioned, regardless of its disputed status.

## 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.[[11]](#footnote-11) 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.[[12]](#footnote-12)

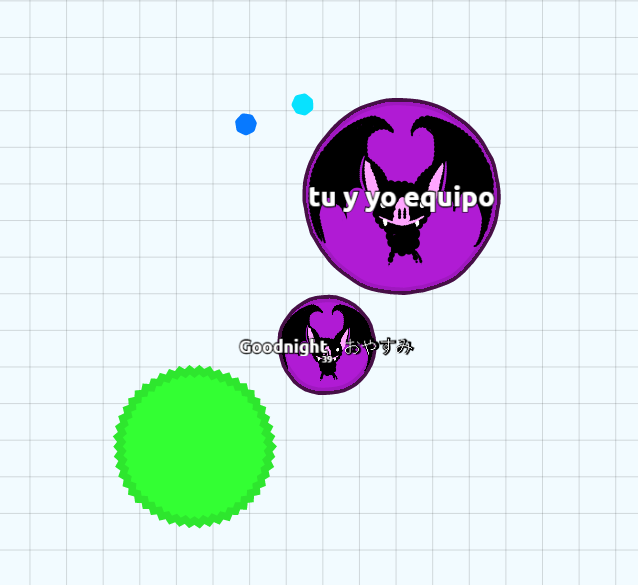


Figure : Agar.io (left) and Auralux (right).

Base building represents the dynamics of players securing the locations of abundant resources by base expansions. Unit building represents the limited selection of units 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 researches and technology upgrades, and the purchases of unique units and items affecting overall gameplay strategies.[[13]](#footnote-13)

These strategies usually involve applying upgrades which helps to make their units perform better than they would expect the performances of their opponents’ units, or playing mind games to deceptively lure their opponents to their downfall.[[14]](#footnote-14) 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 for 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[[15]](#footnote-15), are therefore encouraged to increase the likelihood the player succeeds in completing the game session.



Figure 3: An example of three different levels of veterancy, shown by the number of small white bars above their blue health bar. (GameReplay.org)

## 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 Xn is the increase of a unit upgraded from Level N to Level N+1.

|  |  |  |
| --- | --- | --- |
|  |  |  |

If Xn 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? This main focus of this idea was about experimenting with game balancing 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 balancing.

The custom-built software used a generic unit, represented as a simple, geometric shape, and increases in size as they level up as the game progresses (RTAS). All unit attributes, including permanent bonuses each unit will have as they level up (RTT), are completely customizable to the player beforehand (RTAS). This way, it incorporates unit building (RTS) and veterancy (RTT) into the gameplay. “Veterancy” is a word used to describe the effect of making units with more experiences, such as killing more enemy units or staying alive longer, more effective.[[16]](#footnote-16) This mixing of game elements means the game has multiple genres, but it is still a real-time strategy game since the game elements are either a subgenre or related family of real-time strategy.

## 1.4. Game Balance and Methods to Achieve Balance

Game balance is hard.[[17]](#footnote-17) It is a difficult task where the developers have to approximate the balance of game elements that function completely different from each other, and not to give any players more advantages over the rest. A major reason why game balancing is hard to achieve, is that the game can be difficult to perceive as being balanced.[[18]](#footnote-18) There is no clear, mathematical ways to come up with solutions to make the game balanced, and at the end, the developers would have to make an educated guess the best they can.

Real-time strategy games are notoriously known for their high difficulty when it comes to game balancing.[[19]](#footnote-19) 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 changing their strategies.”[[20]](#footnote-20) 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 in turn, making the game boring to play.[[21]](#footnote-21) Therefore, game developers strive to make their gameplay achieve the Nash equilibrium.

We look into methods of balancing games by having a couple of strategies for the players to choose upon, and allowing as minimal dominant strategies as possible, in order to minimize factors that may hinder the evaluations, and achieve Nash equilibrium. 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 our custom software. *Total War: Shogun 2*[[22]](#footnote-22), *Total War: Attila*[[23]](#footnote-23), and *Multiwinia*[[24]](#footnote-24) are all real-time strategy games where unit compositions on opposing teams 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.



Figure 4: Total War: Shogun 2 gameplay. (3D Juegos)



Figure 5: Total War: Attila gameplay. (Hi-Def Digest)



Figure 6: Multiwinia gameplay. (NewGamerNation)

Games with more complicated unit attributes and geographical properties that affect player decisions, such as be *Starcraft II*[[25]](#footnote-25), *Warcraft III*[[26]](#footnote-26), and *Total Annihilation*[[27]](#footnote-27), requires a more complex game balance. 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]](#footnote-28), and thusly, makes them attractive to the players and motivate them to obtain these upgrades.



Figure 7: StarCraft II gameplay. (Softpedia)

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. Other than environmental obstacles, the game, *Homeword: Deserts of Kharak*[[29]](#footnote-29), encourages the players to see further by advancing to higher ground, and to avoid lower ground near enemies to prevent hostile attacks. Players are given the option to use smoke screens to block the enemies’ line of sight, thus preventing them from receiving excessive fires when retreating from enemy units.

Similarly, some real-time strategy games, such as *Auralux*[[30]](#footnote-30) and *Footmen Wars*[[31]](#footnote-31), 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 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. There has been research done on exploring map layouts and game balance in real-time strategy games, using *Planet Wars*, which is based on *Galcon[[32]](#footnote-32)*, as the basis of their research.[[33]](#footnote-33) What they have found is, as long as the map layout does not provide players more advantages over their opponents, it is considered to be a balanced map.



Figure 8: WarCraft III: The Frozen Throne gameplay (Blizzard)

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.[[34]](#footnote-34) 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 9: Total Annihilation gameplay. (Wayward Strategist)



Figure 10: Homeworld: Deserts of Kharak gameplay. (Scientific Gamer)



Figure 11: Footman Wars gameplay. It is a custom map played in WarCraft III: The Frozen Throne. (Blizzard)

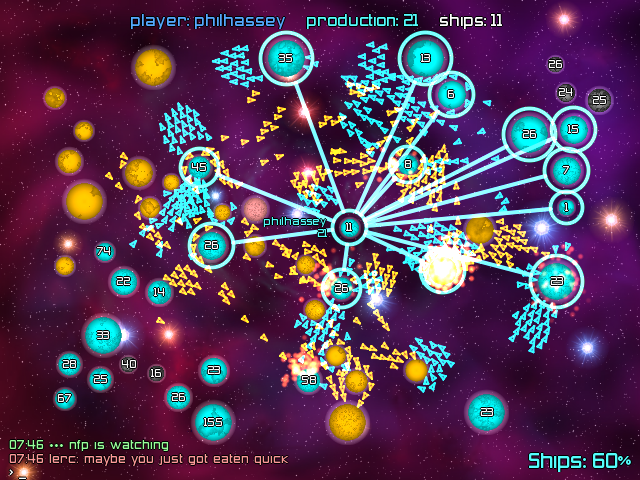


Figure 12: Galcon gameplay (Phil Hassey).

## 1.5. Game Feature and Appeal

A game is defined as “a system of rules in which agents compete by making ambiguous decisions.”[[35]](#footnote-35) We group the system of rules to be a part of the game as features that the players can interact with. Thus, the game features become unique sets of system of rules that make up the game. When ambiguous decisions have meanings and repercussions within the game system, it has been said to cause new challenges to emerge, and will have an impact on the final outcome of the game.

The impact may be a certain type of experience the players may find attractive, or the players may perceive a liking for the experiences.[[36]](#footnote-36) This attraction is defined as the appeal to the game, or the perceived notion of the game’s appeal to the players. In short, by giving the ambiguous decisions of tweaking the game balance through the use of the unit attributes editor, we can then evaluate the editor, as a game feature, if it can increase the game appeal. Since game balance itself is a very debatable game design aspect in real-time strategy games, its meaning is therefore perceived to be very appealing, due to how ambiguous the decisions of approaches for tweaking the game are.

# 2. Game Design

## 2.1. Overview

In this section, the game players and the tool users are all described as “end users”, and the tool version and the game version are referred to as a “software”.

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 messing around with the game, exploring its limitations, creating and pursuing personal objectives, and other motivations.[[37]](#footnote-37) 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 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.

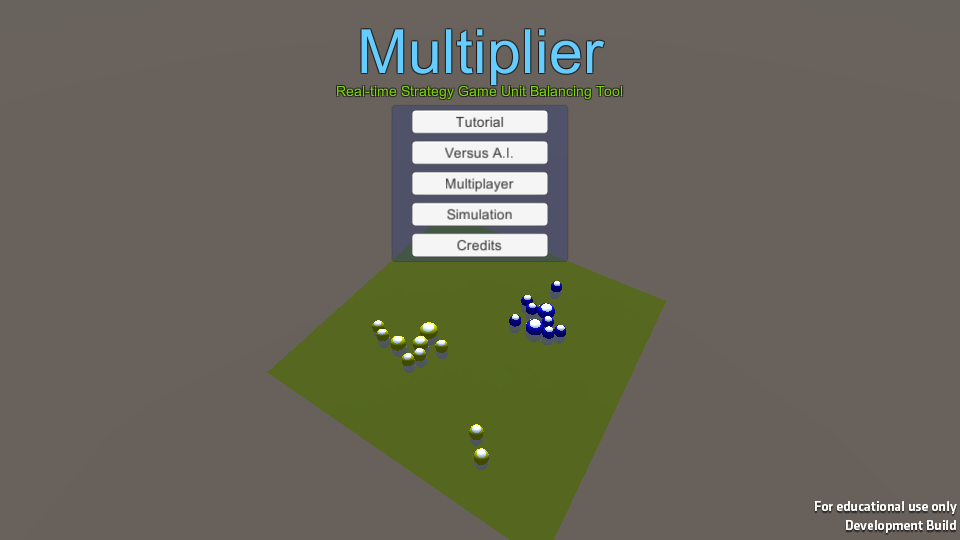


Figure 13: Multiplier, the real-time strategy unit balancing tool, a.k.a., the “Software - Tool version”.

The software contains a variety of game modes, which are Singleplayer, Multiplayer, and Simulation. Only the tool version software build allows the end users to test and tweak the unit attributes, so that the end users can verify if the game units are balanced enough. If balanced, the end users may choose to apply the mathematical equation to their own games as their heuristics for a balanced unit leveling progression. For the game version software build, the end users are able to play Singleplayer 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 defines 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.[[38]](#footnote-38) From a general point of view, defining elements are: resource management, base building, and enemy annihilation.[[39]](#footnote-39) Optional elements include stressing the importance of micromanagement and macromanagement, complicated unit interactions, and tactical strategies players can choose to put in practice. [[40]](#footnote-40) 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*[[41]](#footnote-41), 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.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Using the above equation, game units would be exactly twice more powerful when upgraded, and continues 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 math equations that are still simple to remember, but adds a bit of complexity to it overall. These equations are given as follows:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

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 feature consists of many UI components, creating an elaborate interface used for tweaking the numbers and values. This is done by having the end users provide a math equation, and then generate a series of results for each level, reaching up to Level 10. The end users can provide math equations by typing into the Equation Input Field. Using the Shunting Yard Algorithm[[42]](#footnote-42), the provided equation is then parsed and calculated to correctly print out the results.

Due to the many UI components possibly confusing the end users, toggleable tooltips have been applied everywhere to provide hints and tips on how to use them to the end users.

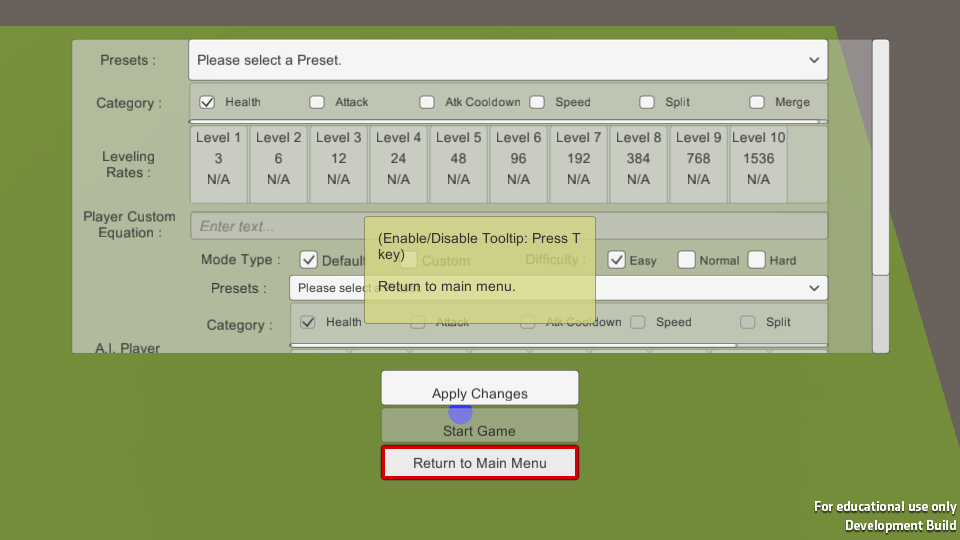


Figure 14: The Attributes Editor for Singleplayer Mode, with the Tooltip and the corresponding indication box shown.

### 2.3.2. Splitting and Merging

The main game mechanics are game unit splitting and merging. 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).

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.

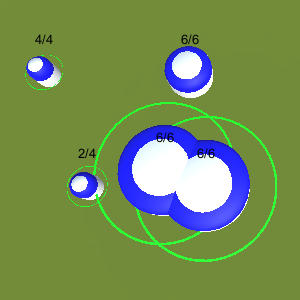


Figure 15: Splitting (left) and Merging (right).

### 2.3.3. Minimap

Inspired by the camera panning movement from WarCraft 3[[43]](#footnote-43), 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.



Figure 16: Minimap and camera boundaries. Note: Text cannot be removed in Unity 5 Education and will always be present.

### 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[[44]](#footnote-44), is known in the military, referring to the general friction of forces, which are danger, exertion, uncertainty, and chance, comprising the climate of war. 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. In other words, only the players can see objects within their line of sight. 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.

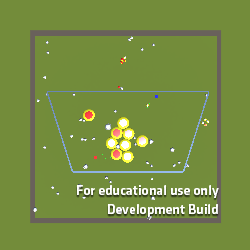


Figure 17: The minimap shown without any fog-of-war. Some of the units are taking damage and attacking opponents.

### 2.3.6. A.I. Gameplay

This feature can easily be observed in Simulation Mode, and is playable in Singleplayer 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 18: Simulation Mode shown in the Tool build.

These features of gameplay are what makes 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 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, while we only provided Level Difficulty in the Game version.



Figure 19: Multiplayer Mode, showing the LAN options, Attributes Editor and the Level Difficulty settings. The editor will only appear when the player chooses the “Custom” preset option.

## 2.4. How Unit Attributes Relate to Game Design

To explain, the 6 main unit attributes used in the software is applicable to most types of gameplay. They are built upon game elements from gathering mechanics in real-time strategy games to form correlations that apply. Here are a few examples:

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.

There are many 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 at.
* 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, did initial work had begun. 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.

Throughout the transitional phases from Unity 5.2.0f1 to 5.3.4f1, the project had encountered many hurdles, some of which can be said as “broken” or “unusable” for a brief time. In the end, the project has successfully completed its course, and development has reached its initial goals.

# 4. Evaluation

This study is done to measure the game appeal of a technical game feature, and to see if it is possible to raise a game appeal based on such feature alone. It does not consider other factors, such as how the experiences of the user interface were received, how simple the game art is, how the game was designed, how useful the tool was, what uses the tool can be applied to, and so forth. Because game appeal is a very subjective concept of how likeness the players perceive when playing games, and since it really depends on how the game was designed, it is ill-advised to consider the result as an overall outlook for most tools to appeal to gamers.

## 4.1. Research Method

Once we have 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 is a total of 51 participants involved in this experiment. 1 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 finished completing their session, they are then given the post-test survey.

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. Research Question

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? |

## 4.3. Result

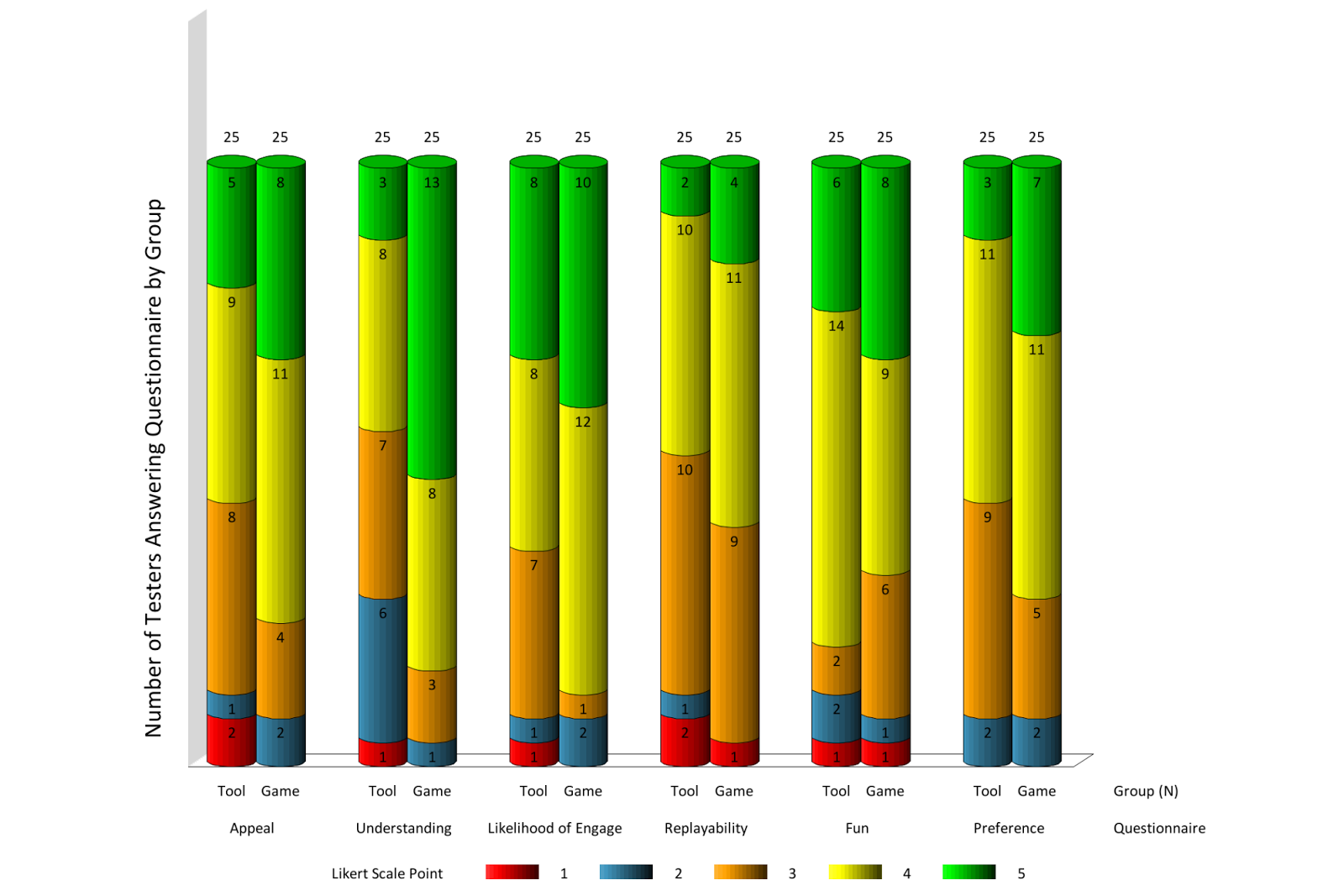
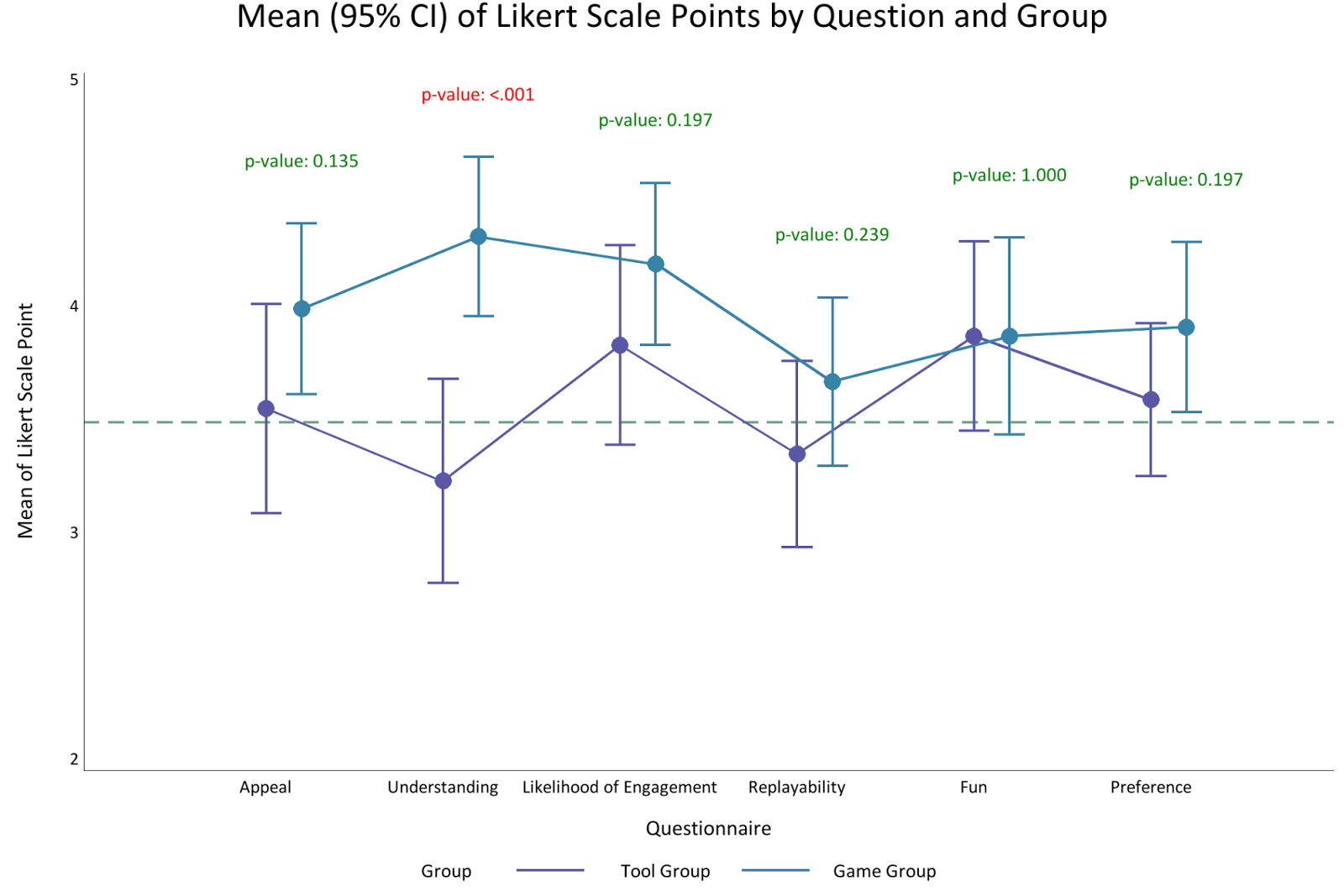


Figure 20: Graphical interpretation of the results of the 6 5-point Likert scale questions from the post-test survey.

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 graph 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.

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. 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.

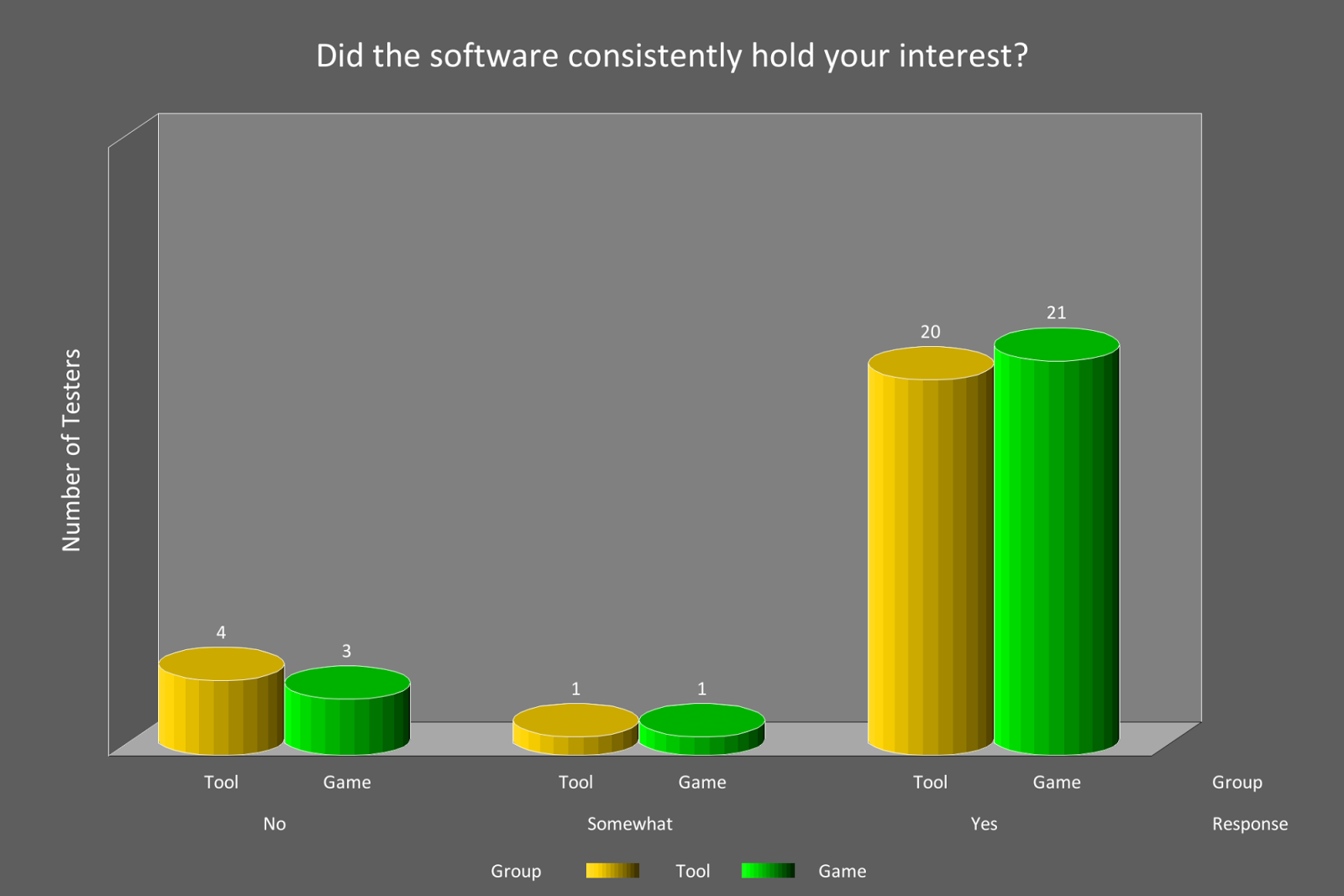
Figure 21: Mean of Likert Scale Points ordered by Questions and Group, with 95% confidence interval.

In the above graph 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.

For the Fun category, 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.

We looked into the Feedback open-ended questions, checking to make sure if the Appeal really poses little to no significance. The results are given in the following figure.

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

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

# 5. Conclusion

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. We feel if there are more participants involved, we may be able to draw a better conclusion.

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 the software used for both the Tool group and the Game group, 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. This abundance of text may have put off and distracted the end users.

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

The following section suggests ideas and plans for the continuation of this project, if there is more time for further research.

## 6.1. Software Improvements

Software development is an ongoing iterative process of improving the software to make it reach its highest potential. With the feedback gathered from the post-test survey, there are many necessary changes needed to address. 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 the demand for it is required.

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 in the case of ours.[[45]](#footnote-45)

## 6.2. Procedural Content Generation

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.[[46]](#footnote-46) It has also been proven that it is possible to have automated content generation in mainstream games.[[47]](#footnote-47) Notable examples include *Minecraft*[[48]](#footnote-48) and *Mini Metro*[[49]](#footnote-49), in which the former uses procedural content generation to generate terrain, and the latter uses procedural audio generation.



Figure 23: Procedural Terrain Generation in the game, Minecraft. (Mojang)

Procedural content generation in real-time strategy games is one of the most interesting challenges in the video game development process.[[50]](#footnote-50) 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, of which procedural map generation for a real-time strategy game has already been explored[[51]](#footnote-51).

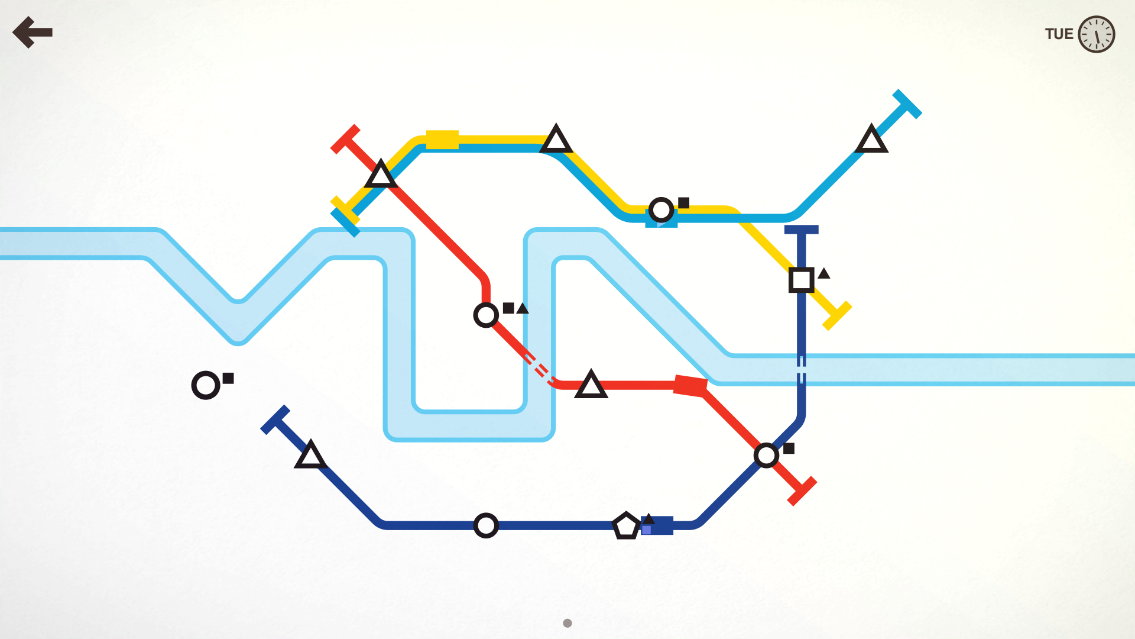


Figure 24: Mini Metro gameplay. This game uses procedural audio generation for its game audio. (Dinosaur Polo Club)

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.[[52]](#footnote-52) Other researches involve using procedural map generations built to fulfill requirements in order to maintain interesting and appealing games, suggesting that game balancing can be perfectly achieved only on extremely dull games.[[53]](#footnote-53) It also theorizes having moderate dynamics and moderate balancing can give ample stimuli to players to expand and to seek their enemies.

# 7. Postmortem

This section explains the woes and troubles the author has to go through, and shares his experiences to interested readers. It is written in first-person perspective.

## 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.

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 know from the beginning that I only have 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 again.

Instead of doing a rewrite due to time constraints, I have to resort to static animations and force the players to read monotonic 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 have 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 to the players 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.

### 7.2.2. 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 starts to get confusing.

When doing an objective comparison, the tester needs to know what things are selected for comparison, and they are testing those things out beforehand. Once the testers did their playtesting sessions, they 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.

Upon rethinking about how I would test, I could have done a double-blind 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. Or maybe there could be a randomness factor added in, so all testers are testing out a software build that was randomly chosen from the two builds for them to test first, before the other is tested.

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.

## 7.3 What did I learn?

### 7.3.1. Things do not always go the way you wanted 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.[[54]](#footnote-54) 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.

# 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

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/

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

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.

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.

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

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.

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/

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

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/

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

The first appendix should be the IRB questionnaires handed to the testers/volunteers. This is important. Can be split up into multiple appendices in case 1 section is not enough.

Any other following sections can have anything I want that is related to this whole project, even if it is irrelevant but useful resources. This includes project documents, documentation and API manuals, game art, conceptual designs, irrelevant sketches, notes, scrawls, etc.

Proposals can also be added into the appendices, but it must be marked as old, new, or anything in between. (Uncertain if that is the case.)

Nothing stops me from adding nothing, though.

1. (Pennington, 2003) [↑](#footnote-ref-1)
2. (Nutt, 2014) [↑](#footnote-ref-2)
3. (Xiong & Iida, 2014) [↑](#footnote-ref-3)
4. (Cheng & Thawonmas, 2004) [↑](#footnote-ref-4)
5. (International Abstract Games Organization, 2010) [↑](#footnote-ref-5)
6. (Thompson, 2000) [↑](#footnote-ref-6)
7. (Miniclip, 2016) [↑](#footnote-ref-7)
8. (War Drum Studios, 2016) [↑](#footnote-ref-8)
9. (Hale-Evans, 2007) [↑](#footnote-ref-9)
10. (Looney Labs, 2016) [↑](#footnote-ref-10)
11. (Walker, 2004) [↑](#footnote-ref-11)
12. (Kleinberg, 2011) [↑](#footnote-ref-12)
13. (Giant Bomb, 2016). Macromanagement is derived from micromanagement in real-time strategy games. [↑](#footnote-ref-13)
14. (Lahiri, 2010) [↑](#footnote-ref-14)
15. (YurdleTheTurtle, 2010) [↑](#footnote-ref-15)
16. (Wikia, 2014), (Wikia, 2016), and (OpenRA, 2016) [↑](#footnote-ref-16)
17. (Burgun, Understanding Balance in Video Games, 2011) [↑](#footnote-ref-17)
18. (Burgun, Understanding Balance in Video Games, 2011) [↑](#footnote-ref-18)
19. (Egenfeldt-Nielsen, Smith, & Tosca, 2012) [↑](#footnote-ref-19)
20. (Nash, 1950) [↑](#footnote-ref-20)
21. (Egenfeldt-Nielsen, Smith, & Tosca, 2012) [↑](#footnote-ref-21)
22. (Onyett, Total War: Shogun 2 Review, 2011) [↑](#footnote-ref-22)
23. (Hafer, 2015) [↑](#footnote-ref-23)
24. (Griliopoulos, 2008) [↑](#footnote-ref-24)
25. (Blizzard Entertainment, 2015) [↑](#footnote-ref-25)
26. (Blizzard Entertainment, 2002) [↑](#footnote-ref-26)
27. (Dulin, 1997) [↑](#footnote-ref-27)
28. (Bangay & Makin, 2013) [↑](#footnote-ref-28)
29. (Blackbird Interactive, 2016) [↑](#footnote-ref-29)
30. A minimalistic real-time strategy game for Android, based in outer space. (Parker, 2013) [↑](#footnote-ref-30)
31. 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) [↑](#footnote-ref-31)
32. *Galcon* is a real-time strategy game where you conquer planets by invading with your ships. (Hassey, 2016) [↑](#footnote-ref-32)
33. (Lara-Cabrera, Cotta, & Fernández-Leiva, A Procedural Balanced Map Generator with Self-adaptive Complexity for the Real-Time Strategy Game Planet Wars, 2013) [↑](#footnote-ref-33)
34. Unit interaction is discussed in the Introduction of (Li Yan, 2014). [↑](#footnote-ref-34)
35. (Burgun, What Makes a Game?, 2012) [↑](#footnote-ref-35)
36. Perceiving a liking to a game feature is done by justifying the choice to obtain a higher level of entertainment. See (Slovic, 1995) in regards to choices and the prominence effect. [↑](#footnote-ref-36)
37. (Breslin, 2009) [↑](#footnote-ref-37)
38. (Ontañón, et al., 2013) [↑](#footnote-ref-38)
39. 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. [↑](#footnote-ref-39)
40. 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. [↑](#footnote-ref-40)
41. (StrategyWiki, 2014) [↑](#footnote-ref-41)
42. (Dijkstra, 1961) [↑](#footnote-ref-42)
43. (Blizzard Entertainment, 2002) [↑](#footnote-ref-43)
44. (Shepherd III, 1997). This paper also goes on to define what the term, “fog of war,” is about. [↑](#footnote-ref-44)
45. (Andersen, et al., 2012) [↑](#footnote-ref-45)
46. (Mark Hendrikx, 2013) [↑](#footnote-ref-46)
47. (Hastings, Guha, Member, IEEE, & Stanley, 2009) [↑](#footnote-ref-47)
48. (Gallegos, 2011) [↑](#footnote-ref-48)
49. (Dinosaur Polo Club, 2015) [↑](#footnote-ref-49)
50. (Lara-Cabrera, Nogueira-Collazo, Cotta, & Fernández-Leiva, 2015) [↑](#footnote-ref-50)
51. (Lara-Cabrera, Cotta, & Fern´andez-Leiva, 2012) [↑](#footnote-ref-51)
52. (Fayard, 2007) [↑](#footnote-ref-52)
53. (Lara-Cabrera, Cotta, & Fernández-Leiva, On balance and dynamism in procedural content generation with self-adaptive evolutionary algorithms, 2014) [↑](#footnote-ref-53)
54. (StackExchange, 2016) [↑](#footnote-ref-54)