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

[**No table of figures entries found.** 3](#_Toc446893731)

[Abstract 4](#_Toc446893732)

[Acknowledgements 5](#_Toc446893733)

[Introduction 6](#_Toc446893734)

[Strategy vs. Tactics 6](#_Toc446893735)

[Inspiration 6](#_Toc446893736)

[Procedural Content Generation 7](#_Toc446893737)

[Game Balance and Related Works 7](#_Toc446893738)

[Game Feature and Appeal 9](#_Toc446893739)

[Game Design 11](#_Toc446893740)

[Overview 11](#_Toc446893741)

[Game Mechanics 11](#_Toc446893742)

[Other Things Worth Mentioning 11](#_Toc446893743)

[Resources 12](#_Toc446893744)

[Tools 13](#_Toc446893745)

[Evaluation 14](#_Toc446893746)

[Research Method 14](#_Toc446893747)

[Research Question 14](#_Toc446893748)

[Result / Conclusion 14](#_Toc446893749)

[Postmortem 15](#_Toc446893750)

[What went right? 15](#_Toc446893751)

[What went wrong? 15](#_Toc446893752)

[What did I learn? 15](#_Toc446893753)

[Conclusion 16](#_Toc446893754)

[Future Work 17](#_Toc446893755)

[References 18](#_Toc446893756)

[Appendices 22](#_Toc446893757)

# Abstract

Video games are a medium of boundless potential when it comes to game features. Developers can create anything, define any rules, and engage the players in many ways, all to make the players feel entertained to continue playing. These are the reasons why gamers love video games, because they can have so much in them.[[1]](#footnote-1) And because of these nearly limitless constraints, it is hard to figure out what game features that entertains the gamers to include when making video games.

This paper aims to look at one particular game feature, an editor for customizable game unit attributes, and evaluates whether having this feature is appealing or not to the players. To evaluate this, two versions of a custom real-time strategy game are built. One version has the editor enabled to the players, and the other version has the editor disabled.

Once both versions of the software are completed, a total of 50 testers were asked to try out the software. The testers are split into 2 groups, the Tool group, which tests the version with the editor enabled, and the Game group, which tests the version with the editor disabled.

The results from the groups show the version with the editor enabled is less appealing than the version with the editor disabled.

Note that the custom-built game can be used to explore the premises of game balance using a mathematical approach, and whether or not procedural content generation game balance for real-time strategy games is possible. However, this paper does not explore these premises, and it would require further research for related future works.

# 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.

# Introduction

## Terminology

Almost two decades ago, the term “real-time strategy” was defined as “a type of strategy game where it closely resembles reality, in which time is limited, and if the player loses time, their opponents may have already taken advantage of it.”[[2]](#footnote-2) This term was used since the early 1980s, at the time for describing what an action strategy game, *Cosmic Conquest*, plays like in the table of contents of the publication magazine, *BYTE*.[[3]](#footnote-3) However, the cofounder of Westwood Studios, Brett Sperry, is mainly credited for using the term to market their game, *Dune II*.[[4]](#footnote-4) This is what we used to define the “real-time strategy” genre in video games.

Nowadays, “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[[5]](#footnote-5). These games require the players to do real-time planning based on incomplete information and they need to handle resource management themselves.[[6]](#footnote-6)

## Strategy vs. Tactics

Real-time strategy games are sometimes confused with real-time tactics, in terms of game mechanics and gameplay.[[7]](#footnote-7) Real-time tactics is a subgenre or a related genre of real-time strategy games, which removes the aspects of base-building, or in general, reducing the importance of macromanagement. 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.[[8]](#footnote-8)

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.[[9]](#footnote-9) 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.[[10]](#footnote-10)

Real-time tactics, on the other hand, is about the placements of units on the battlefields, the unit troop formations, and the exploitation of terrain and environment for tactical advantages against enemies. 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 is therefore encouraged to increase the likelihood the player succeeds in completing the game session.

## Origins

When evaluating real-time strategy games, it is pretty difficult to say which player is better than the others, when all players do not have any prior knowledge of the situation. We can therefore assume each player have equal, negligible levels of certainty to successfully execute mind games for tricking and deceiving others. This way, we are able to simplify many factors based on player intuition and subjective decisions that other players may or may not dare to play. Instead, we look to evaluating performances based on upgrades being applied to game units, in which the process of improving game unit performances fares with how well a player is when compared to the rest of the players or enemies. This is more apparent when players are confronted with other players with stronger army compositions and higher tiered units.[[11]](#footnote-11)

This leads us to the very core of determining the game balance, the variables of play. 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.

To come up with the best approach 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. Whatever outcome it may be, will pave the way to future works in the realm of procedural content generation in real-time strategy unit balancing.

## Procedural Content Generation

Procedural content generation in real-time strategy games is one of the most interesting challenges in the video game development process.[[12]](#footnote-12) 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[[13]](#footnote-13).

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.[[14]](#footnote-14) It has also been proven that it is possible to have automated content generation in mainstream games.[[15]](#footnote-15) Notable examples include *Minecraft*[[16]](#footnote-16) and *Mini Metro*[[17]](#footnote-17), in which the former uses procedural content generation to generate terrain, and the latter uses procedural audio generation.

## Game Balance and Related Works

Game balance is hard. Period.

It is a difficult task where the developers have to balance game elements that function completely different from each other, and not to give a player more advantages over the others. A major reason why game balancing is hard to achieve, is it can be difficult to perceive the game as being balanced.[[18]](#footnote-18) There is no clear, mathematical ways to do this, and at the end, the developers would have to make an educated guess.

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, developers must carefully test all potential interactions and ensure they are balanced and fair across different types of terrain, maps, game modes, and scenarios. Here, there is a particular interest in the concept of Nash equilibrium[[20]](#footnote-20), and related concepts of dominant strategies, in which there exists an equilibrium state where no players can benefit from changing their strategies. Meaning, players will tend to gravitate towards the most optimum strategy, or the dominant strategy. The existence of such strategy saps away the potential for choice, thus making the game boring to play.[[21]](#footnote-21)

Here, we look into balancing the game by having very few strategies for players to choose from, and allowing as few dominant strategies as possible, in order to minimize 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.

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.[[22]](#footnote-22) 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.[[23]](#footnote-23) 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*[[24]](#footnote-24), *Total War: Attila*[[25]](#footnote-25), and *Multiwinia*[[26]](#footnote-26) 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*[[27]](#footnote-27), *Warcraft III*[[28]](#footnote-28), and *Total Annihilation*[[29]](#footnote-29). 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[[30]](#footnote-30).

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*[[31]](#footnote-31), which encourages the players to see further with higher ground, and to avoid lower ground from enemy fires, the player is given the option to use smoke screens to block enemy line of sight, thus preventing players from receiving excessive fires when retreating from enemy units.

Similarly, there are some real-time strategy games, such as *Auralux*[[32]](#footnote-32), which utilizes map layouts designed with a blend of *Footmen Wars*[[33]](#footnote-33)in mind. Research has been done exploring map layout and balance in real-time strategy games[[34]](#footnote-34), 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 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.

You can start to see many varieties of ways to approach game balance in real-time strategy games, but most of all, unit interaction is one of many core components of real-time strategy games.[[35]](#footnote-35) It is because of this, experimenting the possibilities of game balancing using mathematical equations is the main focus of this research project.

## Game Feature and Appeal

As quoted, a game may be defined as “a system of rules in which agents compete by making ambiguous decisions.[[36]](#footnote-36)” We group the system of rules to be a part of the game as features the players interact with. Thus, game features are unique sets of system of rules that make up the game. When ambiguous decisions have meanings and repercussions within the game system, it is 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.[[37]](#footnote-37) 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 a meaning to the unit attributes editor, we can then define the editor as a game feature with the appeal of being able to manipulate game unit attributes. 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.

# Game Design

## Overview

In this section, the game players and the tool users are all described as end users.

The entire software is structured as a tool where the end users are able to interact with the editor and tweak the properties of the given 6 unit attributes used in the game. The editor allows the end users to define any possible leveling progressions, or power-ups, using mathematical equations. This also means the power-ups can be curved down into the negatives if the end users wished so.

A real-time strategy game is purposefully built in a way where game units are splitting and merging itself. 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 to merge. Merging units will upgrade their units to the next level, at the cost of a second unit of the same level prior to merging. As the end users continue to split and merge units, they will reach a state where neither player will win, or will win after a certainly long period of time.

With the editor and game, the end users are able to play a variety of game modes, which are Singleplayer, Multiplayer, and Simulation. All of these game modes are for 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.

## History

The original premise of the software was a game designed around the possibility that complex unit interactions is defined using mathematical equations. Not to be burden with how complex the mathematical representation was going to be, as well as the technical limitations to accomplish this, starting from very simple equations was a better starting point to begin with. 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 Equation (1), 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.

## Game Mechanics

The following list contains game mechanics included in the entire software as features:

* Attributes Editor
* Game Unit Splitting / Merging.
* Minimap
* Autonomous Unit Attacks
* Lack of Fog-of-War
* A.I. gameplay

The first main game mechanic is the editor feature, in which the real-time strategy game is applied to. 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 providing hints and tips on how to use them.

The second main features of 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. 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.

Next feature in the list is the Minimap feature. 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.

Next feature is unit attack automation. 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.

This next one, the lack of fog-of-war, comes with a caveat as part of the software. 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. Here in this software, it lacks the fog-of-war element, so the end users are expected to look at unit interactions and observed if their units are balanced. This is the caveat that differs from typical real-time strategy games, and may limit itself on how truly accurate it portrays a gameplay session.

Finally, there is the A.I. gameplay, which 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.

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

## 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 consist of elements from generic real-time strategy games to form correlations. 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.

# Resources

The main resources used is Unity, where the project uses Unity Networking, or UNET for short. It can be spotty sometimes, and bugs/glitches tends to pop up from time to time, but UNET is very reliable when it comes to internet connection stability.

Other resources include the following:

* Unity3D Forums, where professionals, amateurs, students, and hobbyists come and enjoy discussions on various matters.
* Reddit, where it is mostly showing off concepts of ideas that the game, or me, can witness and discuss about.
* Freenode IRC, where you get to know a few people working on other things and helping each other out.

# Tools

This section discusses the use of Unity, and all resources related to Unity, Unity Networking, shortcomings and issues with Unity, and other advice worth sharing.

# Evaluation

The meat and grits of this paper. State the project’s goal, and come up with the hypothesis that goes into evaluating the project to be successful or failure. Subsections must go into detail of how the evaluation is done, and so forth.

## Research Method

## Research Question

## Result / Conclusion

# Postmortem

Typical game postmortem structure goes here. For reference, see Gamasutra postmortems.

## What went right?

## What went wrong?

### 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.

## What did I learn?

# Conclusion

This section contains the final evaluated answer to the hypothesis stated in the Evaluation section. Shortcomings of this project is also included in this section. Make sure limitations are noted. Never gimp out on the details.

# Future Work

State the endless possibilities this project could have, assuming there are no deadlines and unlimited budgets are given. State when this project is deemed complete, and state what possible research can be made and for what other purposes.

Uncertain if this section should contain Github project repository links, and explanation on how to use the project.

# 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

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

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/

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.

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

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

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

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

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/

# 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. (Nutt, 2014) [↑](#footnote-ref-1)
2. (Byte Publications, 1982) [↑](#footnote-ref-2)
3. (Byte Publications, 1982) [↑](#footnote-ref-3)
4. (Geryk, 2001) [↑](#footnote-ref-4)
5. (Xiong & Iida, 2014) [↑](#footnote-ref-5)
6. (Cheng & Thawonmas, 2004) [↑](#footnote-ref-6)
7. (Walker, 2004) [↑](#footnote-ref-7)
8. (Giant Bomb, 2016). Macromanagement is derived from micromanagement in real-time strategy games. [↑](#footnote-ref-8)
9. (Kleinberg, 2011) [↑](#footnote-ref-9)
10. (Lahiri, 2010) [↑](#footnote-ref-10)
11. (Blizzard Entertainment, 2009) [↑](#footnote-ref-11)
12. (Lara-Cabrera, Nogueira-Collazo, Cotta, & Fernández-Leiva, 2015) [↑](#footnote-ref-12)
13. (Lara-Cabrera, Cotta, & Fern´andez-Leiva, 2012) [↑](#footnote-ref-13)
14. (Mark Hendrikx, 2013) [↑](#footnote-ref-14)
15. (Hastings, Guha, Member, IEEE, & Stanley, 2009) [↑](#footnote-ref-15)
16. (Gallegos, 2011) [↑](#footnote-ref-16)
17. (Dinosaur Polo Club, 2015) [↑](#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. (Fayard, 2007) [↑](#footnote-ref-22)
23. (Lara-Cabrera, Cotta, & Fernández-Leiva, On balance and dynamism in procedural content generation with self-adaptive evolutionary algorithms, 2014) [↑](#footnote-ref-23)
24. (Onyett, Total War: Shogun 2 Review, 2011) [↑](#footnote-ref-24)
25. (Hafer, 2015) [↑](#footnote-ref-25)
26. (Griliopoulos, 2008) [↑](#footnote-ref-26)
27. (Blizzard Entertainment, 2015) [↑](#footnote-ref-27)
28. (Blizzard Entertainment, 2002) [↑](#footnote-ref-28)
29. (Dulin, 1997) [↑](#footnote-ref-29)
30. (Bangay & Makin, 2013) [↑](#footnote-ref-30)
31. (Blackbird Interactive, 2016) [↑](#footnote-ref-31)
32. A minimalistic real-time strategy game for Android, based in outer space. (Parker, 2013) [↑](#footnote-ref-32)
33. 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-33)
34. (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-34)
35. Unit interaction is discussed in the Introduction of (Li Yan, 2014). [↑](#footnote-ref-35)
36. (Burgun, What Makes a Game?, 2012) [↑](#footnote-ref-36)
37. 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-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)