Virus Buster:: Unity

Solo Group Cephus

Final Report

CS 467 Capstone Project
8 / 18 / 2017

by
James Palmer

I . Introduction

Midway through this term, there was a group change. In result, I was replaced into my own solo group, Team Cephus. The group project goal is to make a single level tower defense game, in 18 days. I choose to use Unity game engine with the intention to create a general Tower Defense game while adding a few personal twists. The idea behind my game, Virus Buster, is to defend your own personal computer from being attacked by 3 different viruses sent by another computer.

Personally, this was a daunting task at first. Going into this project, I had zero experience with unity, as well as all, all other assistance programs I choose for this project. I found Unity to be a fun experience. Primarily, once one updates any Inspector changes in code, you can see the results almost immediately. More so, I can have control up to a given single frame. Beyond that, it was a challenge to learn Unity specific coding that comes with C#.

For simplicity sake, a tower defense game is as follows. A player defends a goal from opposing enemy attackers. One does this by creating towers, guns or allies to attack these enemies who march along a directed path. For complete definition, see below:

> *Tower defense (TD) is a subgenre of strategy video game where the goal is to defend a player's territories or possessions by obstructing the enemy attackers, usually achieved by placing defensive structures on or along their path of attack. This typically means building a variety of different structures that serve to automatically block, impede, attack or destroy enemies. Tower defense is seen as a subgenre of real-time strategy video games, due to its real-time origins,though many modern tower defense games include aspects of turn-based strategy. Strategic choice and positioning of defensive elements is an essential strategy of the genre. -**Wikipedia Definition***

It’s dangerous to fight Viruses alone, take these!

II. Setup & Usage

Executable Game Application

Virus Blaster game is pre-compiled into an non-downloadable webpage:

https://returningdawn.itch.io/virus

which runs with Google Chrome. A Unity-Launcher loading screen will appear and you can play your game from that browser. Please be patient, the game takes up to a few minutes to appear, when loading for the first time.

***Controls*:**
[W-S-A-D] for movement
[ESC or P] for pause menu
[Space] to lock camera
[Mouse Wheel] to Zoom in and out
[Left Mouse Button] Interact with Objects

- Click turret icon
- Click location on map to place turret
- Click on turret to select or deselect upgrade//sell menu

Below is a short pictorial tutorial video that I wanted to implement. Unity, works with QuickTime directly. However, because I am using a Windows processor AND I cannot afford QuickTime for Windows. It is impossible to integrate a video within the inspector of Unity.
https://vimeo.com/230081754

Brief video of my friend, Ember Dahl, testing my game. Noted, before upgraded enemies were implemented AND she failed to beat the game during her first attempt.
https://vimeo.com/230101794

III. Software Systems - Unity Overview

This project relied almost exclusively on unity as our core Software System which is connected directly with Visual Studios. If one would like to play Virus Buster from the Unity game console or want to learn about Unity UI, read below. If one doesn't have interest in Unity UI, please skip to section **IV Sprites, Prefabs and Overlays**.

v Download and Understanding Unity

1. Download/install latest version of Unity Game Engine at https://unity3d.com/
2. If Unity asks to make an account, just make a free personal use one
3. Extract a desired repository and link with your Unity Game Engine.
4. Open said project repository folder in unity as a project

Understanding the Layout and functions of Unity

Scene - The scene contains the objects of the game. Each unique Scene file acts as their own level. This is where you place unity scene definitions, environments, obstacles, and decorations, essentially designing and building your game in pieces. See https://docs.unity3d.com/Manual/UnityHotkeys.html to help easily navigate through your Scene.

Hierarchy - On the left of the screen is the Project "Hierarchy". The Hierarchy window contains a list of every GameObject in the current Scene. Some of these are direct instances to Files, Prefabs and custom created objects. As objects are added and removed from your Scene, they are also appear and disappear from your Hierarchy.

Inspector - The Unity Inspector contains scripts, sounds, Meshes and other graphical elements. Here in the Inspector, all information is displayed in detailed fashioned about current state of the selected GameObject. The Inspector also allows users to modify all the functionality of your GameObjects in your Scene.

GameObjects - The GameObject is the most important concept in Unity. Everything in Unity is a GameObject, from characters, to lights, cameras and special effects. These GameObjects are just sprites by themselves. As a user, you add properties to the GameObject to give it movement, character or special effects.

Project Window - This is or editor window which holds all of our files. Everything that is anything in our project is saved here. The User can make new files in the Project Window which proper names. This allows the user to navigate saved information for future use in the Scene. In short, this saves GameObject information such as character models, character color and characteristics.

Toolbar- Bar at the top of Unity which consists of seven basic controls.

- **Transform Tools -** to move, shape, enlarge GameObjects
- **Transform Gizmo Toggles -** This affects the Scene View Display
- **Play/Pause/Step Buttons -** Use to start, pause or step our game

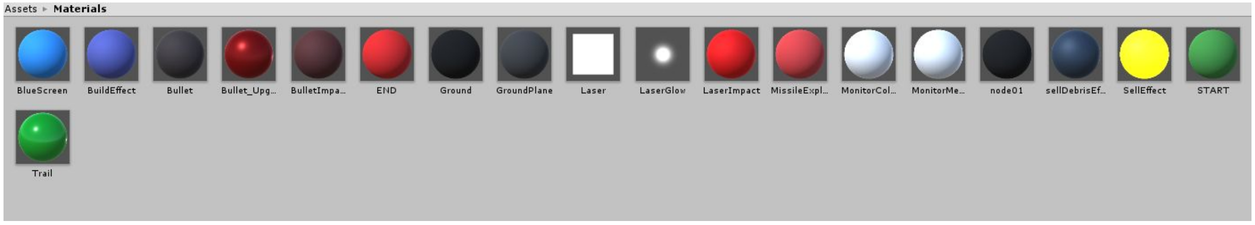
- **Cloud Button -** Opens Unity Service Window
- **Account -** Used to access your Unity Account
- **Layers Drop-Down Menu -** Controls which objects are displayed in Scene view, controls arrangements of all views and controls order.

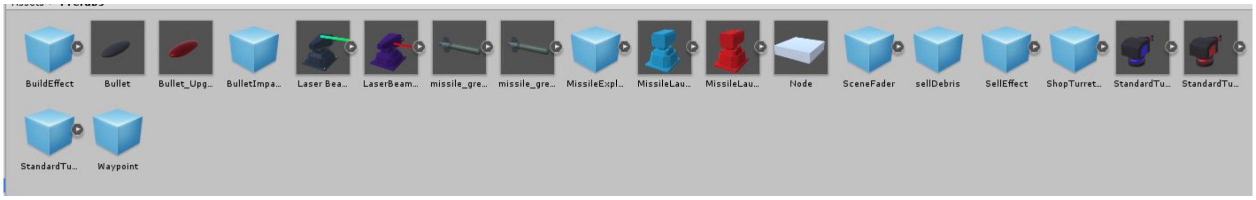
Console - Like most IDEs this is where warnings, errors, or general debug statements. Console logs can be obtained by adding Debug.Log("text here"); into your code.

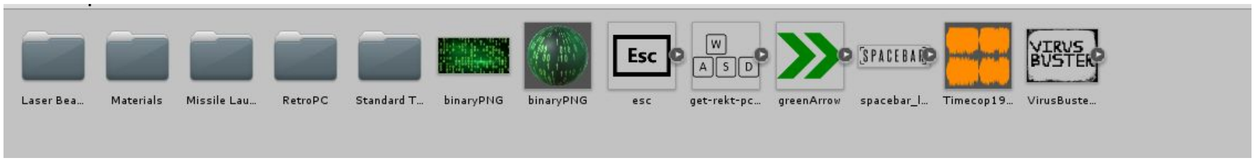
Animator - The animator component is used to assign amination to each GameObject in your Scene. This requires you to reference the Animator Controller which defines which Animations you wish to use. From there, the Animation tool is used to change timings, add movement or characteristics to your GameObject.

Game- This is the cream of the crop, where all your hard work is shown. The Game is where the created game can run without having to create an executable. See Toolbar above for Game basic controls.

IV. Sprites, PreFabs and Overlays



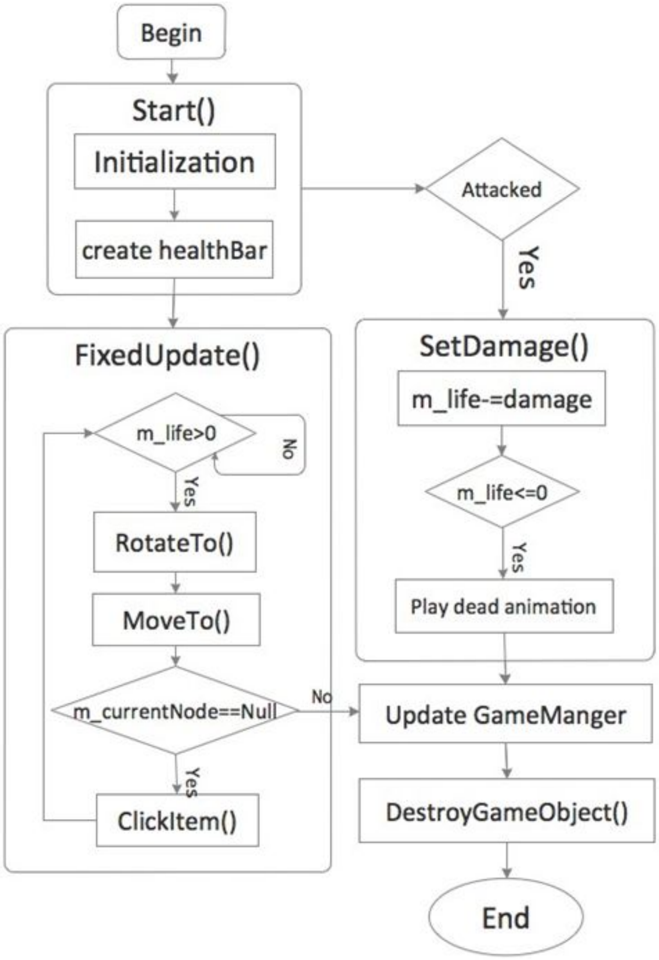




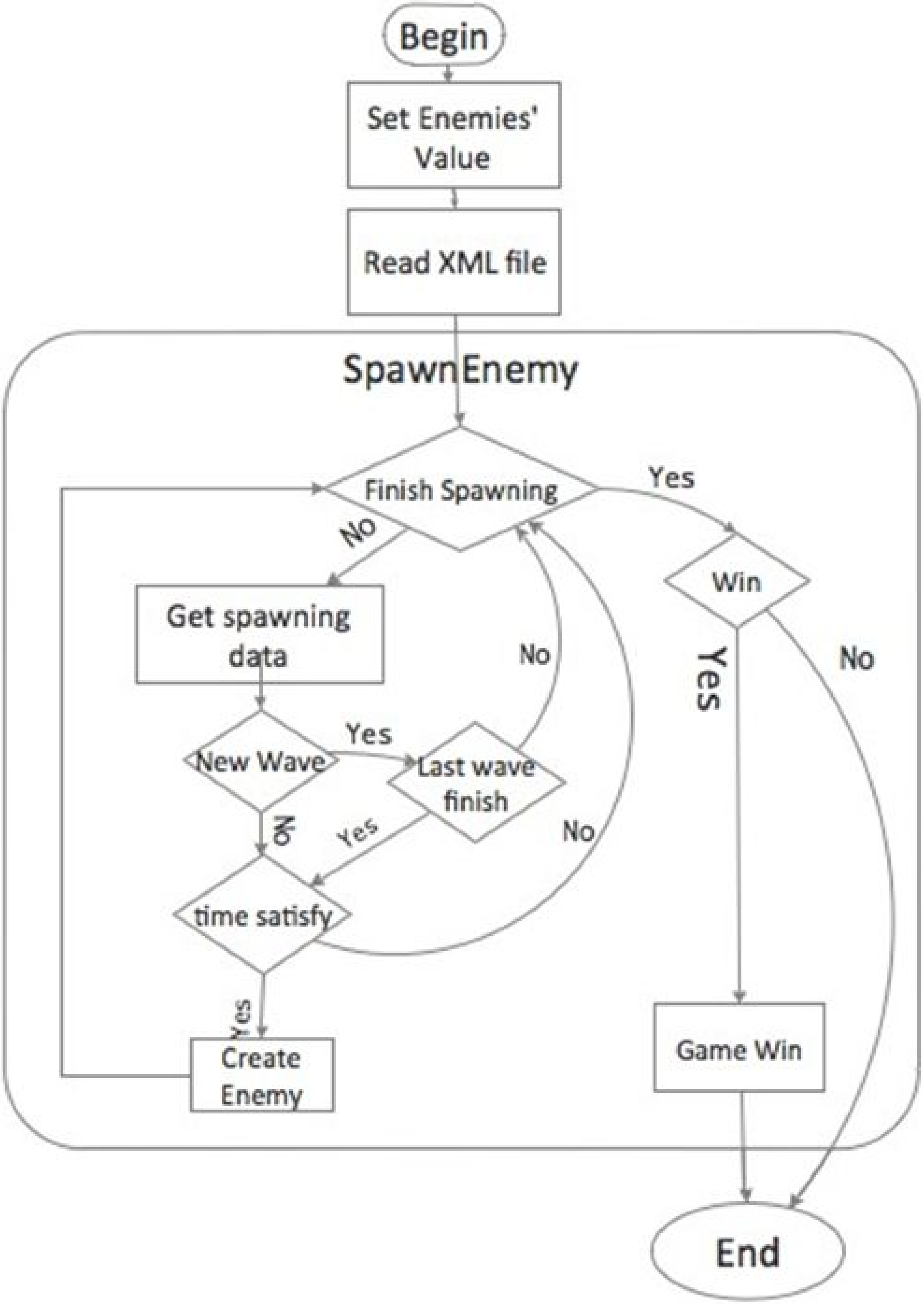
V. Game hierarchy and Balancing Waves

First, how is the game working functionally. Note below, how the game begins, interacts and ends. Noted in section, Project Changes, create healthBar has been removed for frame rate issues.

First photo, is the node hierarchy for how the GameObjects and Scene interacts with the game.

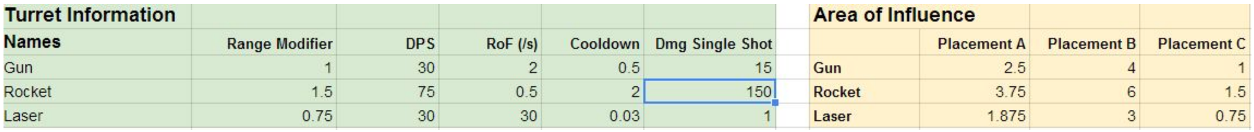


Second photo depicts how the GameManager interacts with the GameObjects and the Scene.


Begin
Set Enemies' Value
Read XML file
SpawnEnemy
Finish Spawning
Yes
NO
Get spawning data
New Wave
Yes
Last wave finish
No
No
Yes
No
time satisfy
Yes
Create Enemy
Win
Yes
No
Game Win
End

Balancing Waves- How the game overlay is designed and Scene overlay is used to determined characteristics of turrets.

Turret information seen in Excel.



Turret Information					
Names	Range Modifier	DPS	RoF (/s)	Cooldown	Dmg Single Shot
Gun	1	30	2	0.5	15
Rocket	1.5	75	0.5	2	150
Laser	0.75	30	30	0.03	1

Area of Influence			
	Placement A	Placement B	Placement C
Gun	2.5	4	1
Rocket	3.75	6	1.5
Laser	1.875	3	0.75

Take into account how often a turret hits an enemy from spot A. Spot A is three spots from our Spawn point. This is important to factor how many times it takes an enemy to be hit until death, how many times it is possible to be hit, then average how many per wave. This helps with bullet fire rate.

Gun Placement A -> Area of Impact = 2.5
Gun's Cooldown = 0.4
Speed of Enemy = 10
Formula = (AOI/Speed)/Cool Down
(2.5/1) / 0.5 = 5
At placement A the gun will hit the Enemy 5 times.

Then, following assessments are taken into account for sequencing through waves.

Of course you can also make this range based (to save you from defining each round separately).

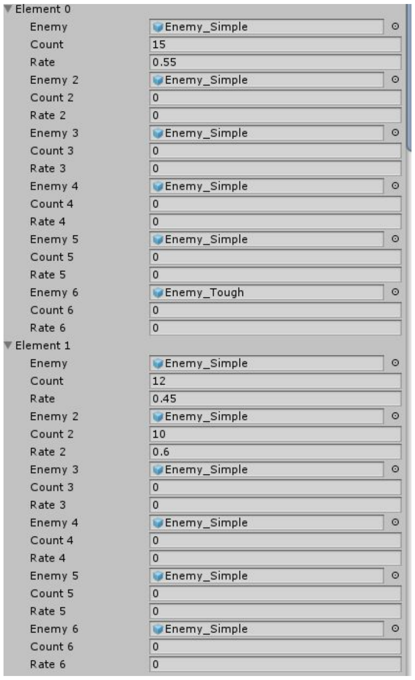
- round 01 - 3: Spawn 10 x enemyA and 5 x enemyB
- round 4 - 6: Spawn 10 x enemyB and 5 x enemyC.

However, this type of ranged-based spawning won't provide much variation. Another approach would be based on probabilities (or weights). Something like:

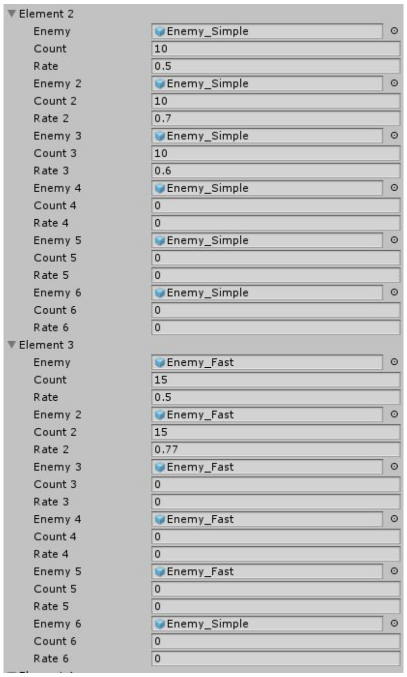
- enemyA (simple) has speed 1.0 which they spawn at round 1, 0.5 at round 3 and possibly 0.3 at round 7
- enemyB (fast) has speed 1.0 which they spawn at round 4, .05 at round 6 and 0.3 at round 9

Then you interpolate the speed given the current round. So with the above definition, enemy A could have a spawn rate of .25 upon the final round. From these ideas, it is more of tinkering each wave rather than equations. How long you want each wave to be, worth decrease depending on wave/game length, etc. After many play tests and a few outside testing sources, below is the final wave output. The waves information is stored into an array. Why you can see that Element starts at 0 and ends at 9 (9 == wave 10). Enemy Prefab is stored, count is amount of enemies that will be released that round and rate. Below is the wavespawner equations followed by the final spawning output.

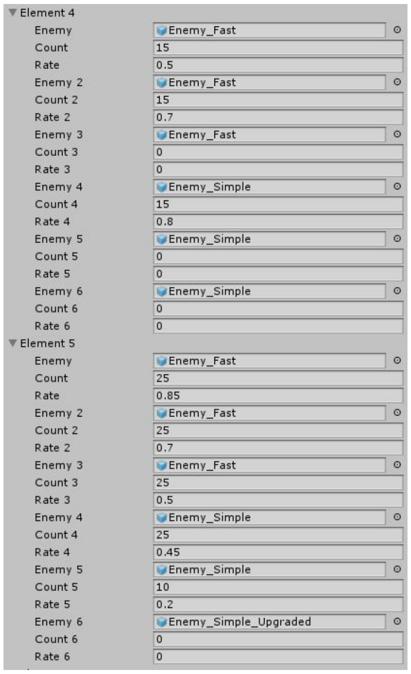
1second / givenRate = wavespawnerRate
1 / .5 = 2 seconds



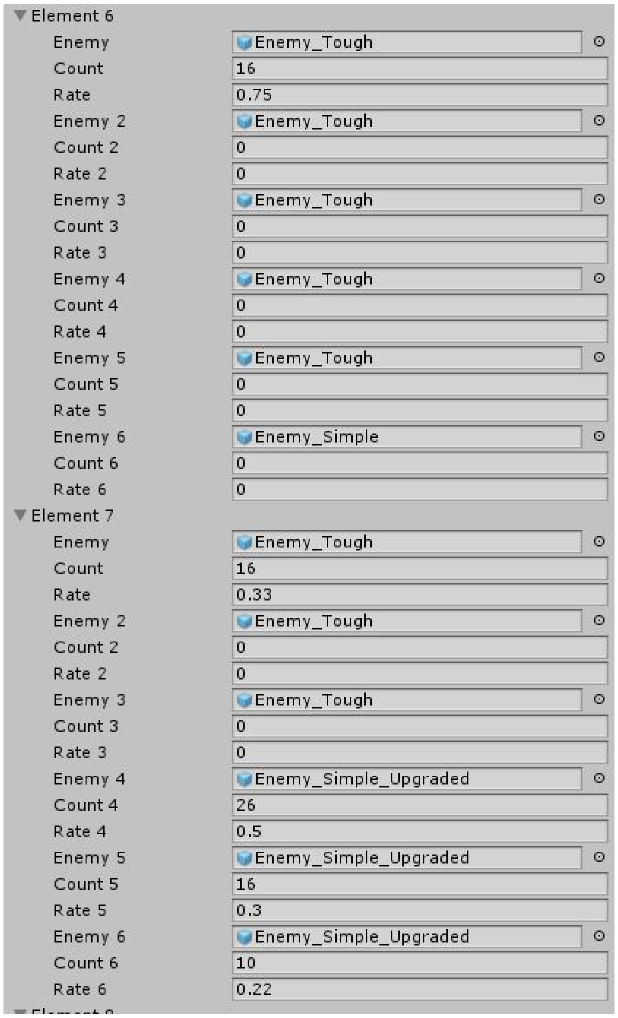
▼ Element 0	
Enemy	Enemy_Simple ⊙
Count	15
Rate	0.55
Enemy 2	Enemy_Simple ⊙
Count 2	0
Rate 2	0
Enemy 3	Enemy_Simple ⊙
Count 3	0
Rate 3	0
Enemy 4	Enemy_Simple ⊙
Count 4	0
Rate 4	0
Enemy 5	Enemy_Simple ⊙
Count 5	0
Rate 5	0
Enemy 6	Enemy_Tough ⊙
Count 6	0
Rate 6	0
▼ Element 1	
Enemy	Enemy_Simple ⊙
Count	12
Rate	0.45
Enemy 2	Enemy_Simple ⊙
Count 2	10
Rate 2	0.6
Enemy 3	Enemy_Simple ⊙
Count 3	0
Rate 3	0
Enemy 4	Enemy_Simple ⊙
Count 4	0
Rate 4	0
Enemy 5	Enemy_Simple ⊙
Count 5	0
Rate 5	0
Enemy 6	Enemy_Simple ⊙
Count 6	0
Rate 6	0



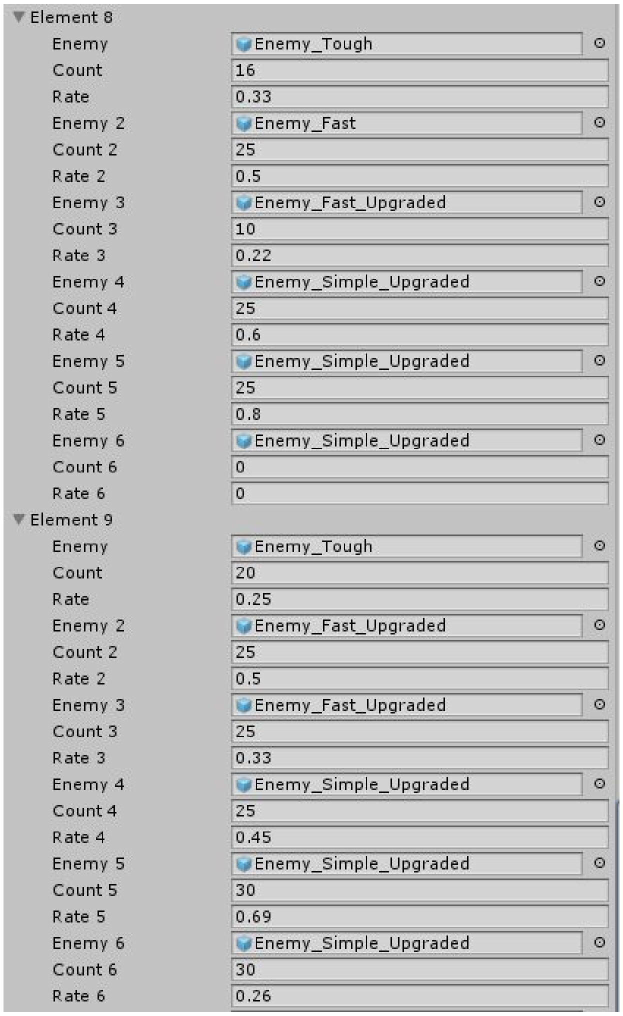
▼ Element 2	
Enemy	Enemy_Simple ⊙
Count	10
Rate	0.5
Enemy 2	Enemy_Simple ⊙
Count 2	10
Rate 2	0.7
Enemy 3	Enemy_Simple ⊙
Count 3	10
Rate 3	0.6
Enemy 4	Enemy_Simple ⊙
Count 4	0
Rate 4	0
Enemy 5	Enemy_Simple ⊙
Count 5	0
Rate 5	0
Enemy 6	Enemy_Simple ⊙
Count 6	0
Rate 6	0
▼ Element 3	
Enemy	Enemy_Fast ⊙
Count	15
Rate	0.5
Enemy 2	Enemy_Fast ⊙
Count 2	15
Rate 2	0.77
Enemy 3	Enemy_Fast ⊙
Count 3	0
Rate 3	0
Enemy 4	Enemy_Fast ⊙
Count 4	0
Rate 4	0
Enemy 5	Enemy_Fast ⊙
Count 5	0
Rate 5	0
Enemy 6	Enemy_Simple ⊙
Count 6	0
Rate 6	0



▼ Element 4	
Enemy	Enemy_Fast ⊙
Count	15
Rate	0.5
Enemy 2	Enemy_Fast ⊙
Count 2	15
Rate 2	0.7
Enemy 3	Enemy_Fast ⊙
Count 3	0
Rate 3	0
Enemy 4	Enemy_Simple ⊙
Count 4	15
Rate 4	0.8
Enemy 5	Enemy_Simple ⊙
Count 5	0
Rate 5	0
Enemy 6	Enemy_Simple ⊙
Count 6	0
Rate 6	0
▼ Element 5	
Enemy	Enemy_Fast ⊙
Count	25
Rate	0.85
Enemy 2	Enemy_Fast ⊙
Count 2	25
Rate 2	0.7
Enemy 3	Enemy_Fast ⊙
Count 3	25
Rate 3	0.5
Enemy 4	Enemy_Simple ⊙
Count 4	25
Rate 4	0.45
Enemy 5	Enemy_Simple ⊙
Count 5	10
Rate 5	0.2
Enemy 6	Enemy_Simple_Upgraded ⊙
Count 6	0
Rate 6	0



▼ Element 6	
Enemy	Enemy_Tough ⊙
Count	16
Rate	0.75
Enemy 2	Enemy_Tough ⊙
Count 2	0
Rate 2	0
Enemy 3	Enemy_Tough ⊙
Count 3	0
Rate 3	0
Enemy 4	Enemy_Tough ⊙
Count 4	0
Rate 4	0
Enemy 5	Enemy_Tough ⊙
Count 5	0
Rate 5	0
Enemy 6	Enemy_Simple ⊙
Count 6	0
Rate 6	0
▼ Element 7	
Enemy	Enemy_Tough ⊙
Count	16
Rate	0.33
Enemy 2	Enemy_Tough ⊙
Count 2	0
Rate 2	0
Enemy 3	Enemy_Tough ⊙
Count 3	0
Rate 3	0
Enemy 4	Enemy_Simple_Upgraded ⊙
Count 4	26
Rate 4	0.5
Enemy 5	Enemy_Simple_Upgraded ⊙
Count 5	16
Rate 5	0.3
Enemy 6	Enemy_Simple_Upgraded ⊙
Count 6	10
Rate 6	0.22



▼ Element 8	
Enemy	Enemy_Tough ⊙
Count	16
Rate	0.33
Enemy 2	Enemy_Fast ⊙
Count 2	25
Rate 2	0.5
Enemy 3	Enemy_Fast_Upgraded ⊙
Count 3	10
Rate 3	0.22
Enemy 4	Enemy_Simple_Upgraded ⊙
Count 4	25
Rate 4	0.6
Enemy 5	Enemy_Simple_Upgraded ⊙
Count 5	25
Rate 5	0.8
Enemy 6	Enemy_Simple_Upgraded ⊙
Count 6	0
Rate 6	0
▼ Element 9	
Enemy	Enemy_Tough ⊙
Count	20
Rate	0.25
Enemy 2	Enemy_Fast_Upgraded ⊙
Count 2	25
Rate 2	0.5
Enemy 3	Enemy_Fast_Upgraded ⊙
Count 3	25
Rate 3	0.33
Enemy 4	Enemy_Simple_Upgraded ⊙
Count 4	25
Rate 4	0.45
Enemy 5	Enemy_Simple_Upgraded ⊙
Count 5	30
Rate 5	0.69
Enemy 6	Enemy_Simple_Upgraded ⊙
Count 6	30
Rate 6	0.26

V. Additional Tools, APIs, Libraries, etc.

Additional Tools used:

Maya- 3D animation, modeling, simulation, and rendering software provides an integrated, powerful toolset. Use it for animation, environments, motion graphics, virtual reality, and character creation.

Tiled2Unity- Unity plugin that allows level specific maps to be imported as a PreFab.

Git and GitHub- Development platform inspired by the way you work. Used to store all code involved in this project. Came in handy when Windows 10 had a large update and dumped my entire project to square one.

Photoshop, BeFunky.com and LunaPic - Online options for photo and Sprite manipulation.

C# programming language with Unity Manual - Zero C# experience going into this project. Luckily, with mass amount of C and C++ experience from this program allowed me to transfer over easily. However, Unity has Unity specific commands. Thanks to Unity Manual and Unity forums, I was able to adapt to the new client.

VI. Project Changes

Half way through the short summer term. My original group was separated and I was placed in a new Solo Group. Because of this, original outline or write up wasn't required. However, that didn't stop some initial notes and ideas found on this google drive to change. These quick changes were decided due to time, skill and simplicity.

Originally, the goal of the game was to have multiple levels, maps and 6 enemy wave runners in a swamp. First problem arose due to the time limit of completing the game in 10 minutes, this beginning parameter was changed to 10 rounds and three enemies. To keep the game interesting, waves were increased in amount of spawn mobs rather than amount of waves.

Second idea was to have a dark and creepy swamp environment. I learned how to use basic Maya, found some pre generated sprites to alter and then apply to my game. First problem stemmed from lack of Maya experience. All of my enemy animation was delayed, lacking or causing errors. Continuation of laughing and error testing. My enemies started to float above set waypoints, which enemies are suppose to navigate towards. Instead, my swamp monsters became floating ghosts whom either died falling off the map or somehow triggered destroyObject(); call from the final waypoint. Due to time issue and lack of knowledge, the idea changed to simple sprites.

Once the enemy sprites were changed to something simplistic, the overall environment needed change as well. With a roll of a dice(yes I rolled a six sided dice to decide an environment). It was decided to have a small binary skin overlay to fall prey to a stereotypical movie perception of the inside of a computer, binary. Each node was changed to a green 1010101 skin with enemies coming from a computer. Nothing is better than different shaped viruses leaving a previously infected computer and attacking your virus free computer.

After a lot of work and multiple outside sources testing the game. Two issues were presented to me. First, there was a large frame rate issue during a large wave. To counteract this, I removed the health bars. Second, the community stated the game was too easy. In response, I upgraded the simple and fast enemy at the end of the game. They are still the same color, however, look more metallic in color and darker. The enemies stats were increased by 1.5. Similar upgrade when upgrading a turret.

VII. Conclusion

Overall this was most enjoyable project of this program. Not only did I learn what is possible with my abilities from this program in less than 20 days, but what is to come as a software engineer. My experience with this project reflects my same experience with this program. It is easy to start but extremely difficult to progress and perfect. From one patch of ground that is mis noted to missing a space or semicolon. Alone, this speaks louder for the difficulty of this major and the future career to come. I learned from this project alone, that creating individual objects with specific scripts have their place but putting in the extra work to make some of the objects usable in other areas makes the overall project more maintainable and will shorten the workload.

Unfortunately, my first group didn’t work out very well. However, this opportunity to do a project on my own was for the best. I only wish that I had more time to tinker, adapt and grow my game to something larger and more refined. In the end, the experience of this project has been very pleasurable. I was able to create an idea from a napkin I wrote on at a bar, to something with personality. Something that stemmed from a concept, online research which then combined with my own idea. I hope you enjoy playing the game as much as I enjoyed creating it. There is no better feeling than seeing something you worked hours on, be enjoyed by others. My two friends whom playtested Virus Buster, will not stop asking about how it came about, when will I update the game and what is next. The excitement, praise and honor is a wonderful feeling and I look forward to what that next thing will bring me.

VIII. Sources

Thirslund, A. (2016). [online] http://brackeys.com/. Available at: http://brackeys.com/ [Accessed 7 Aug. 2017].

Damon Reece. *Best Tower Defense Games of All Time*. http://gameranx.com/features/id/13529/article/best-tower-defense-games/ April 27, 2015 [Accessed 13 Aug. 2017].

Ray Wenderlich. (2015). *How to Create a Tower Defense Game in Unity – Part 1*. [online] Available at: https://www.raywenderlich.com/107525/create-tower-defense-game-unity-part-1 [Accessed 14 Aug. 2017].

Unity Community. (2017). *Manual Index*. [online] Available at: https://docs.unity3d.com/Manual/index.html.1061067/ [Accessed 15 Aug. 2017].

Unity Community. (2017). *TD Wave spawning*. [online] Available at: https://forum.unity3d.com/threads/tower-defense-wave-manager.1061067/ [Accessed 15 Aug. 2017].

Unity Community. (2017). *TD Wave spawning*. [online] Available at: https://forum.unity3d.com/threads/td-wave-spawning.84981/ [Accessed 15 Aug. 2017].

Doucet, L. (2014). *Optimizing Tower Defense for FOCUS and THINKING - Defender's Quest*. [online] Fortress of Doors. Available at: http://www.fortressofdoors.com/optimizing-tower-defense-for-focus-and-thinking-defenders-quest/ [Accessed 15 Aug. 2017].

Bright Lights / Desires. (2015). [Youtube] Available at: https://www.youtube.com/watch?v=Bhkd0877xFo [Accessed 17 Aug. 2017].

Preece, Paul. "Desktop Tower Defense". Handdrawngames.com. September 25, 2008 [Accessed 17 Aug. 2017].

En.wikipedia.org. (2017). *Tower defense*. [online] Available at: https://en.wikipedia.org/wiki/Tower_defense [Accessed 17 Aug. 2017].