5-2019

Grant Anon Minigames Extension

Justin Robbins
justinrobbins@unomaha.edu

Grant Anon:
Honors Minigame Extension

University Honors Capstone

University of Nebraska at Omaha

Justin Robbins
April 26 2019

Advisor:
Dr. Harvey Siy

Abstract

The Grant Anon system was designed to be a casualized version of the real-time strategy genre, a genre usually known for its difficulty and competitiveness because of Starcraft II, the most popular game in the genre. Grant Anon was designed as part of a capstone project, and this report details the extension that was created to add an additional element designed to make it easier for any player to enjoy Grant Anon: minigames. These minigames serve to reduce the skill needed to participate effectively in Grant Anon. This is accomplished by providing an alternative means of gaining an advantage over the other players outside of strategic prowess. The extension not only focuses on adding a single minigame to Grant Anon. The minigame extension creates a structure that allows many more minigames to be easily implemented into the Grant Anon game loop, providing a diverse game experience to widen the game's default audience and provide additional challenges to the players. As a proof of concept, one minigame was created using this structure, called the 'CleanUp.' This report will detail the gameplay of Grant Anon, and how the addition of CleanUp and other minigames could improve the enjoyability of the game as a whole. It will also cover the coded architectural additions added by this extension, and detail the process required to add a new minigame into Grant Anon.

Table of Contents

1 Introduction

Grant Anon was a capstone project created using the Unity game engine by Bréana Townsend, Tristan Martin, and myself, Justin Robbins. We began the project by brainstorming a plethora of possible game features, but ended up cutting many of those functions since we only had a semester to complete the game. For my honors capstone, I decided to take one of the discarded ideas and implement it in addition to the normal work I had to perform for my capstone project. That idea was the idea that adding additional minigames into Grant Anon would diversify the gameplay experience and bring a lot of fun to the user experience to augment Grant Anon's gameplay experience. This idea was inspired by party games such as the *Jackbox Party Box* or *Mario Party* that injected a variety of minigames into their product to greatly vary the type of game the player was experiencing. Obviously I could not match the volume of games that a full development team took years to create, but I could lay the foundation with a skeleton and a proof of concept. Originally I had brainstormed a few different minigames to create, but settled on CleanUp, a minigame centered around cleaning up lab spills caused by a rogue experiment, the Mess Monster. The minigame was built on the ground laid by Grant Anon, so this report will first briefly cover the original game before detailing the work I performed to create the minigame addition structure and the CleanUp game itself, as well as the reasoning followed to justify why Grant Anon needed to have minigames added into it on top of its established gameplay loop.

2 Grant Anon

2.1 Overview

Grant Anon is a real time strategy game, which means that each player is expected to direct an army's strategy and manage their economy in real time, as opposed to having specified turns to execute their strategies (Wayward, 2015). The most significant modern title is Starcraft II, which has an international presence is the Esports scene (Partin, 2018). This gives the game and by extension its genre a reputation as a game of skill, where the competitive scene dominates. Grant Anon was developed as a much more forgiving and casual experience, where friends could play together for fun and not require extreme amounts of expertise to do well in the game. It was also inspired by 'party games' like *Mario Party, Super Smash Bros, Mario Kart*, which focus on couch play so that friends and family can enjoy the game side by side. *Mario Party* in particular has many mini games integrated into it. These minigame are short games contained within the larger game, as minigames often are (Super Happy Games, 2017). In order to accomodate this kind of interaction, Grant Anon was given a local split screen design, where each player needed to connect to the hosting device with a controller in order to play the game with their friends.

2.2 Development

Grant Anon was developed by Tristan Martin, Bréana Townsend, and myself using the Unity Engine, one of the most popular game engines in the industry due to its reputation for ease of

use. Its thorough documentation made it very easy to learn (Unity Technologies, 2019). This allowed the development team to quickly accustom ourselves to the environment and begin developing Grant Anon. This was crucial because of the limited amount of time given to complete the product; the development time was only one semester, and creating any sort of game takes a lot of time and effort. We also used the integrated Visual Studio environment to create C# scripts that created custom behaviors not available in the Unity Editor by default. We used GitHub in order to organize our code and provide version control to ensure the project files were consistent and accessible to each member of the development team. Trello was utilized in order to schedule our process and ensure that each task we set up for development was completed on time, allowing us to efficiently complete development even when we missed our deadlines. Bréanna also used Clip Studio Paint in order to create the Sprite sheet images used to provide Grant Anon's visuals.

2.3 Gameplay

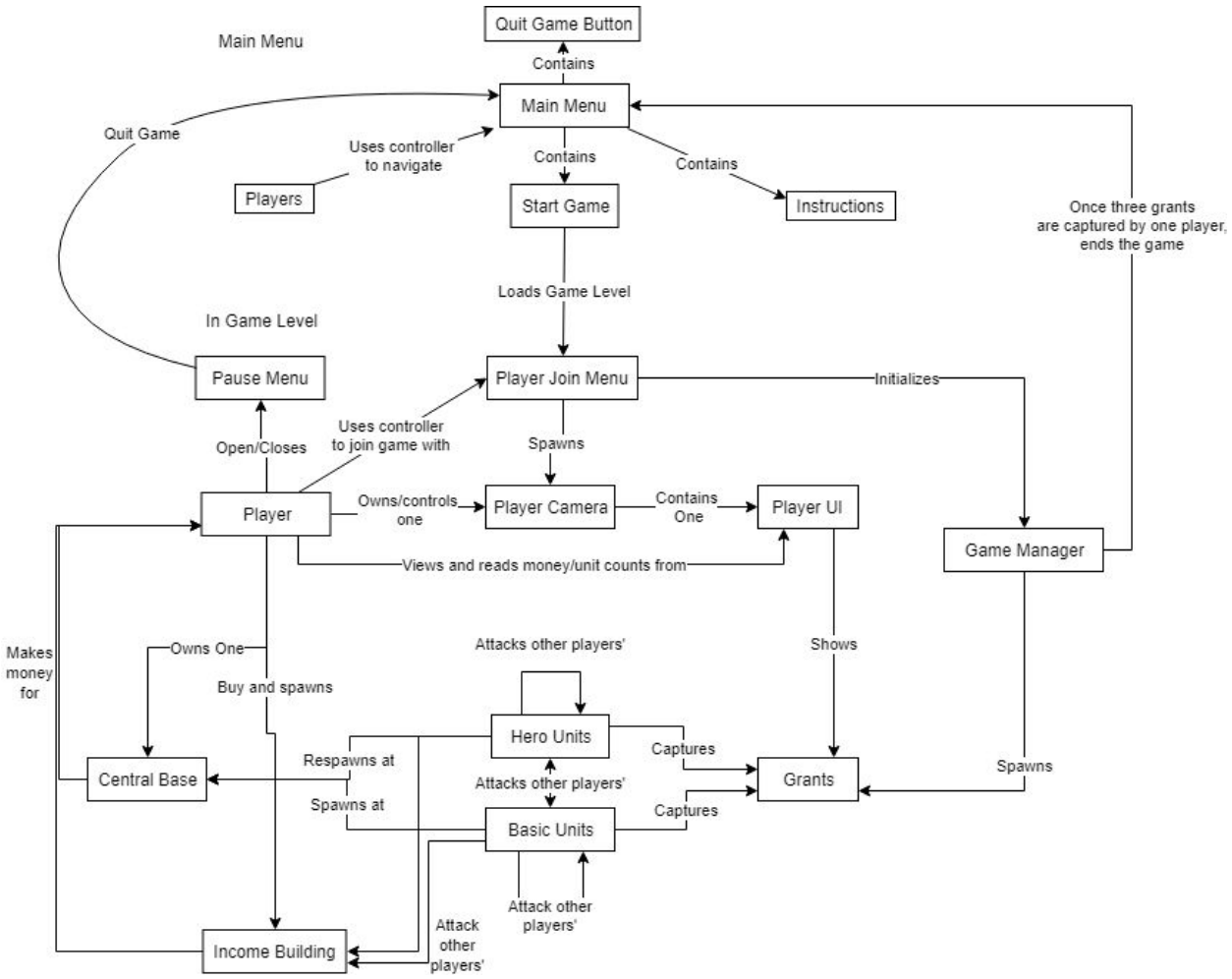

Figure 1.1 - Flow Chart depicting Grant Anon's game flow

Grant Anon's premise is a grim future where colleges are pitched against each other in a blood sport to acquire financial grants, and field teams to compete in 'Grant Anon,' a game where hordes of students and interns battle to capture grants found in the arena. Once one team gets three grants, they win the match. Mechanically, Grant Anon is a 2D isometric real time strategy game that can be played locally with 2-4 players in a single game. A PS4 controller must be connected to the machine running Grant Anon with a micro USB cord. Once the players start the

game, they are asked to join the game in the sign in menu screen. After all players have joined and indicated that they are ready to begin, the game starts. Each player has a designated point on the map where they begin the game, and is spawned into the world with a central base and a hero unit. The central base supplies the player with a slow but steady source of income, while the hero unit is a soldier the player can reliably use to attack and defend other units. It is unique from units the player can buy with money due to its ability to recreate itself for free when it dies after a short timer. These two things supply each player with a basic but reliable way to interact with the game, even if they are at a disadvantage. The steady income provided by the central base can be used to purchase additional units or buildings that give the player additional income. The units are cheaper and provide the player with additional military power. The income buildings are an investment that will allow the player to create more units in the long run. However, these income buildings can be destroyed by other players, so they must be protected with units. This risk and reward dynamic demands that players gauge the threat other players pose to their income buildings and units.

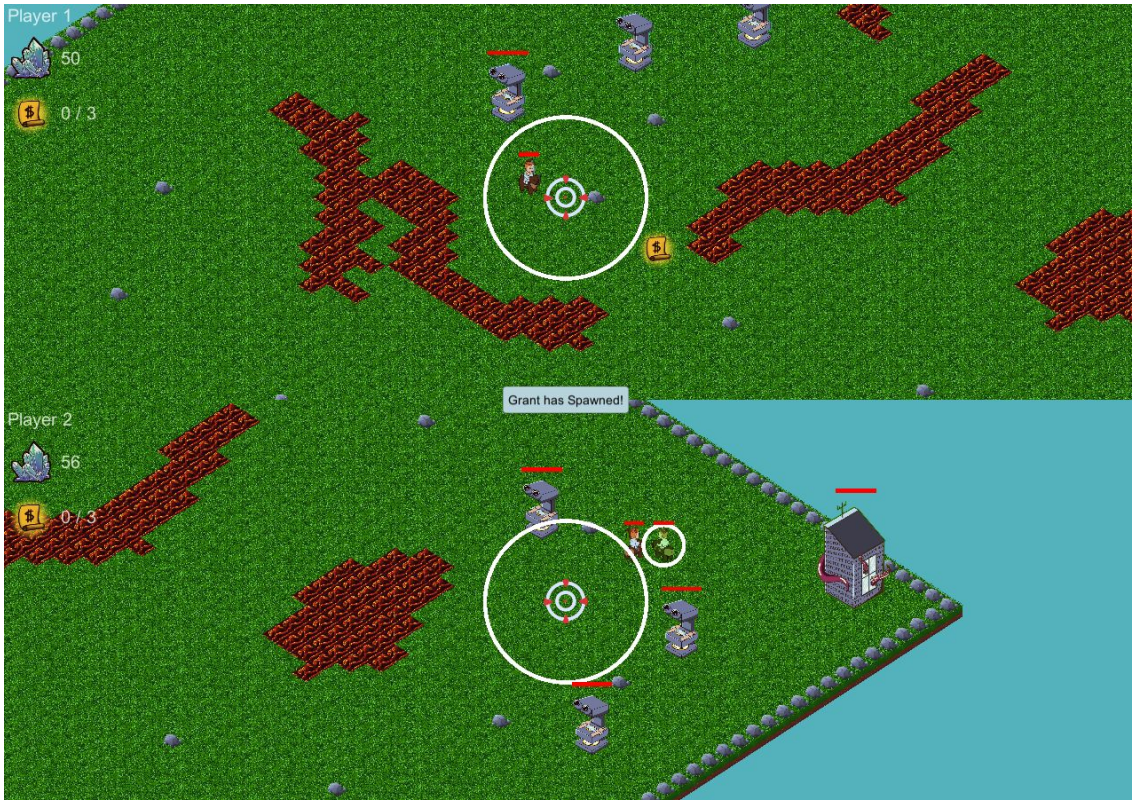

Figure 1.2 - Screenshot of a game of Grant Anon

These buildings and units allow the players to achieve the goal of the game: capturing grants. The middle of the screen contains a timer that counts down the time until the next grant spawns into the game world. Once it does, another will not spawn until one of the players manages to capture the spawned grant. This is done by selecting at least one unit and moving it nearby the grant. The grant will then begin to count down on a ten second timer. If other players' units move into the capture zone, the timer will be paused until only one player's unit is within the grant's capture radius. If the new player is the one remaining, the timer will restart and begin counting down once more. If a single player controls the capture zone for ten seconds, they will score the grant. Once a player scores three grants, they win the game. This process is depicted in the screenshot above, which shows Player 1 in the process of capturing a grant, while Player 2 defends their income buildings.

3 Minigame Extension

3.1 Goals

Grant Anon as developed originally is already a very accessible game due to its simplified nature. However, real time strategy games are innately complicated and may cause people to shy away from the game simply because of that. In order to encourage a wider array to people to play Grant Anon, my honors capstone was to create a way to insert minigames into Grant Anon. The goal of these minigame was to create additional avenues for players to gain advantages in the game outside of pure strategic skill. These minigames require players to use different skills than they would normally have to use in a strategy game, allowing a wider range of players to succeed at Grant Anon through minigames. Winning a minigame provides the winning player with a reward of resources that they can use to create additional units or buildings. These resources can also help players get back on their feet after being defeated by another player in an earlier engagement. Having your buildings and units destroyed by another player would put a player in a huge power disadvantage in normal Grant Anon, but these minigames would allow a disadvantaged player to recover from their defeat. Creating a single minigame would address these issues, but playing a single minigame over and over may get stale over time. To address this issue, I not only created a minigame as a proof of concept, but I also made a general setup that can be used to implement more games into Grant Anon in the future.

3.2 Minigame Structure

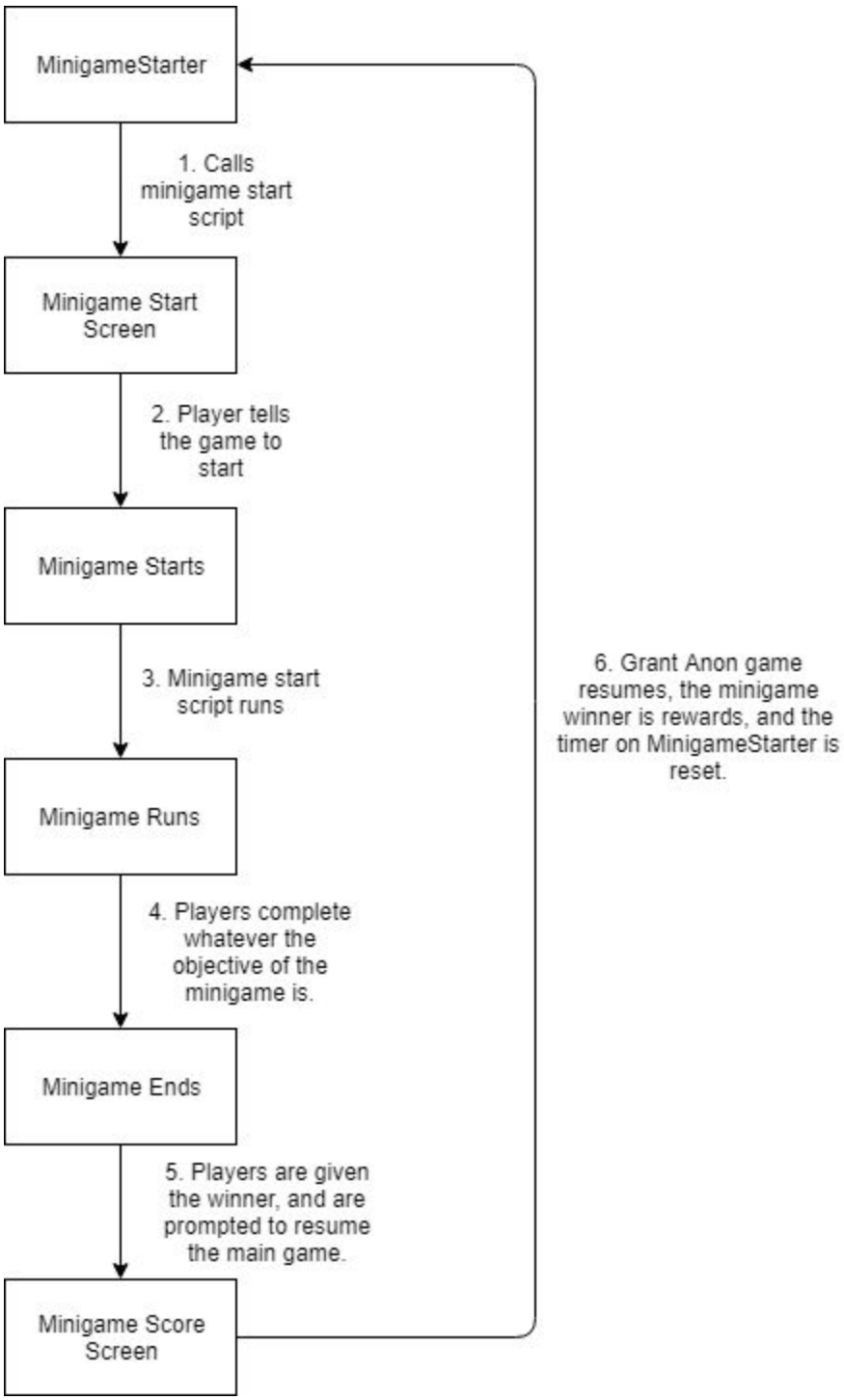

Figure 3.1 - Minigame Implementation Structure

The first goal of this capstone was to create a general design paradigm for adding new minigames into the gameplay loop of Grant Anon. Figure 3.1 depicts the general process a minigame in Grant Anon uses to interrupt and execute. A minigame starter object in the game level counts down a timer until it can be used to activate the minigame. Each minigame starter is associated with another minigame, so that its unique appearance will allow users to know which minigame they are starting when they interact with the starter. When the timer reaches zero, player can move their units near the start object to begin the game. The minigame starter script will then initiate the minigame start sequence by pausing the main Grant Anon game, switching the player cameras to the appropriate minigame cameras, then displaying the minigame start menu. The minigame level itself is found elsewhere in the same scene as the main Grant Anon level, but is otherwise inaccessible to the players. Once the players hit start, the minigame will begin and transition into the main gameplay loop of the minigame. The particular implementation of this section is unique to each minigame and should be implemented as the minigame requires. When the game ends, the end screen will appear and ask the players if they wish to resume the main game once more. Once a player presses start, they will be returned to the main game and Grant Anon will resume as usual. The winning player will then be rewarded with bonus resources that will help them in their efforts to win the game. The start and end menus of the minigames can also be different in order to provide a different experience for each minigame. However, the phases should be followed by every minigame since they need to interact with the Honors Game Manager in the same manner.

Figure 3.2 - CleanUp starter object

3.3 CleanUp Minigame Proof of Concept

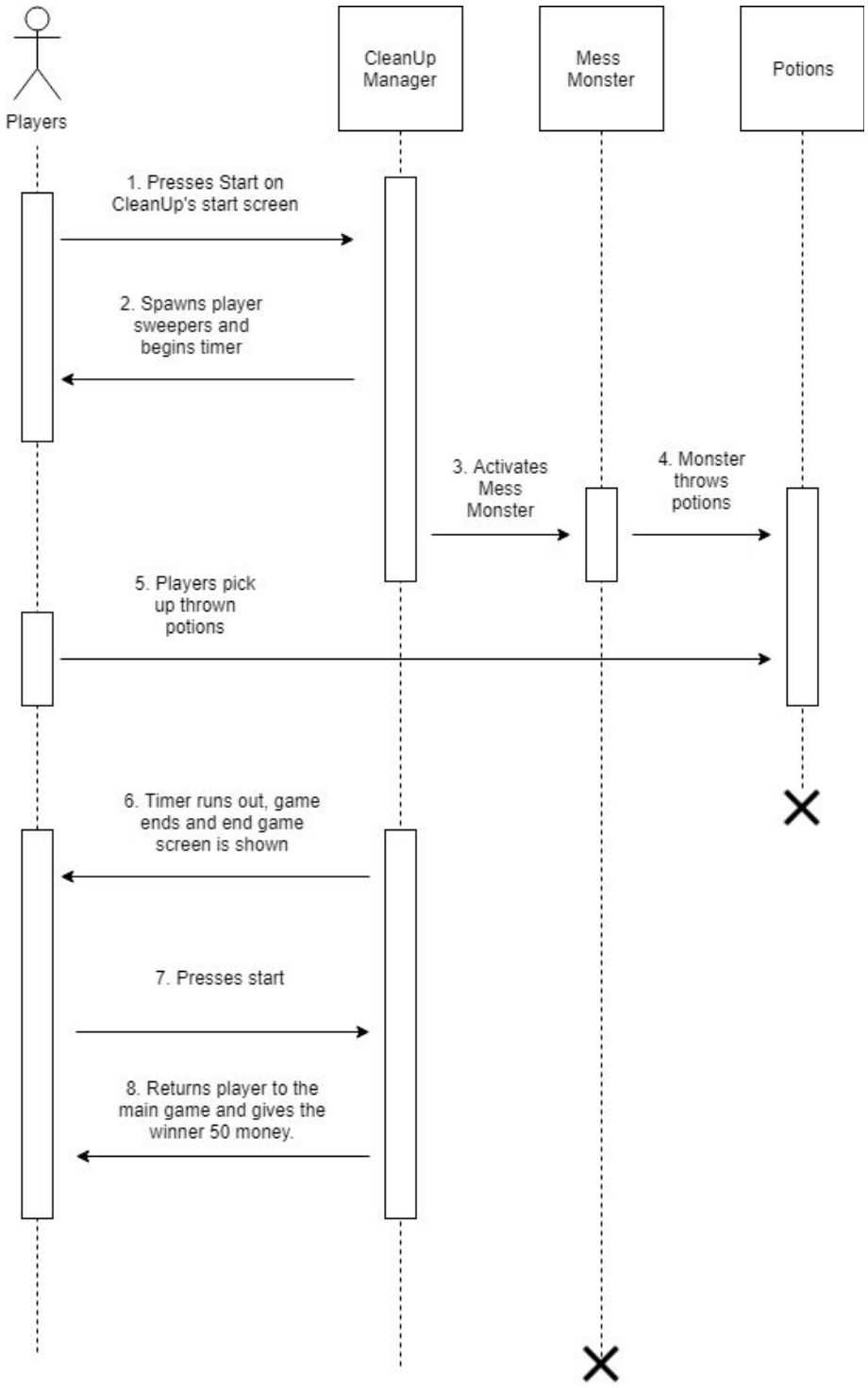

Figure 3.3 - CleanUp Gameplay Loop

The CleanUp minigame was built as a proof of concept of the minigame addition method outlined in the previous section. This minigame is called in the main game through the use of a Minigame Starter object, which called the CleanUpManager script to begin the minigame. This script then calls the GameManager to pause the main game and initialize CleanUp. Then the GameManager pauses Grant Anon and starts the CleanUp game. This loads the CleanUp start screen, which prompts the players to press the start button to begin the game. When any player complies, the script initializes the MessMonster script, which causes the monster at the center of the level to begin slinging potion flasks around the room, which eventually break on the floor of the level as shown in the below picture.

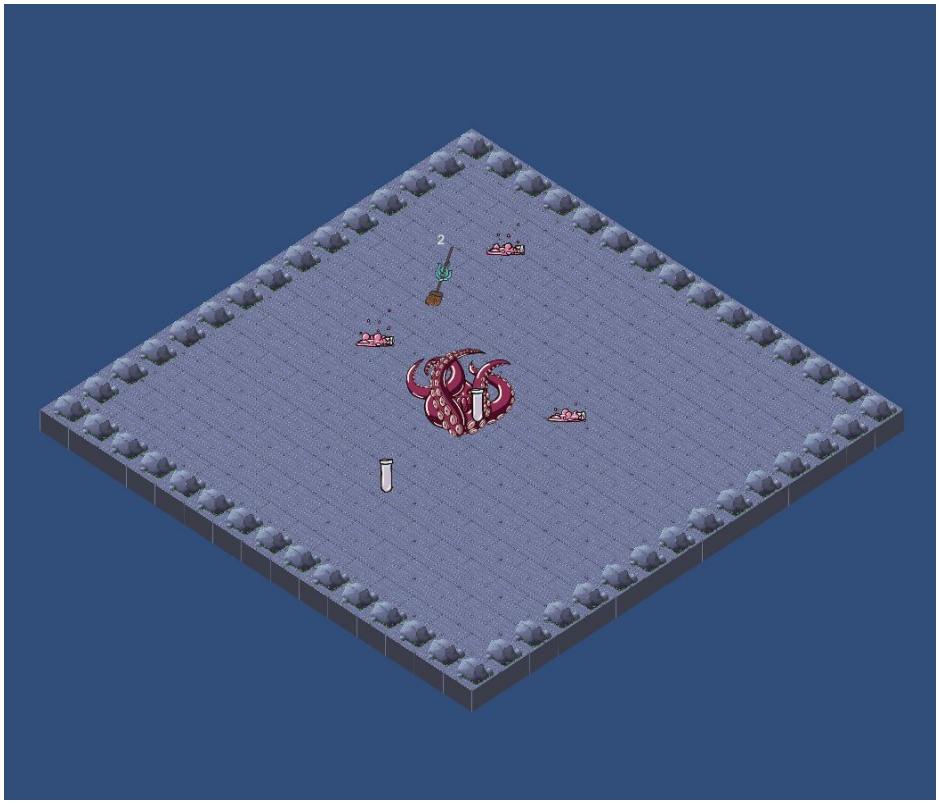

Figure 3.4 - CleanUp Gameplay Screenshot

The CleanUpManager script also spawns in a sweeper object, which looks like a floating broom with a number above it. One sweeper is spawned in for each player, allowing each player to control one of the sweepers. The players then have to move around the level and pick up the potions broken around the level. Whenever a player picks up a potion, their score is incremented and the number above their broom increases to reflect their score. The monster will continue throwing these potions until the level ends, providing continuous potions for the players to collect. The CleanUpManager maintains a timer that represents how long the minigame will last. Once it reaches zero, the minigame ends and the CleanUpManager loads the end menu for the minigame. The players are once again prompted to click start, but this time it's to resume the original game of Grant Anon. These menus are designed to pause the gameplay and allow the players to take a short break if required to get water, food, or use the restroom in game. They also allow players to be more prepared for the sudden shifts in gameplay caused by switching between Grant Anon and the minigames. Once this is done, the CleanUpManager resets the minigame data and calls the GameManager to resume the game and to reward the winning player money. If there was a tie for the winning position, all tied players are rewarded an equal amount of money.

Figure 3.5 - Sweeper

4 Implementation

4.1 Created Files

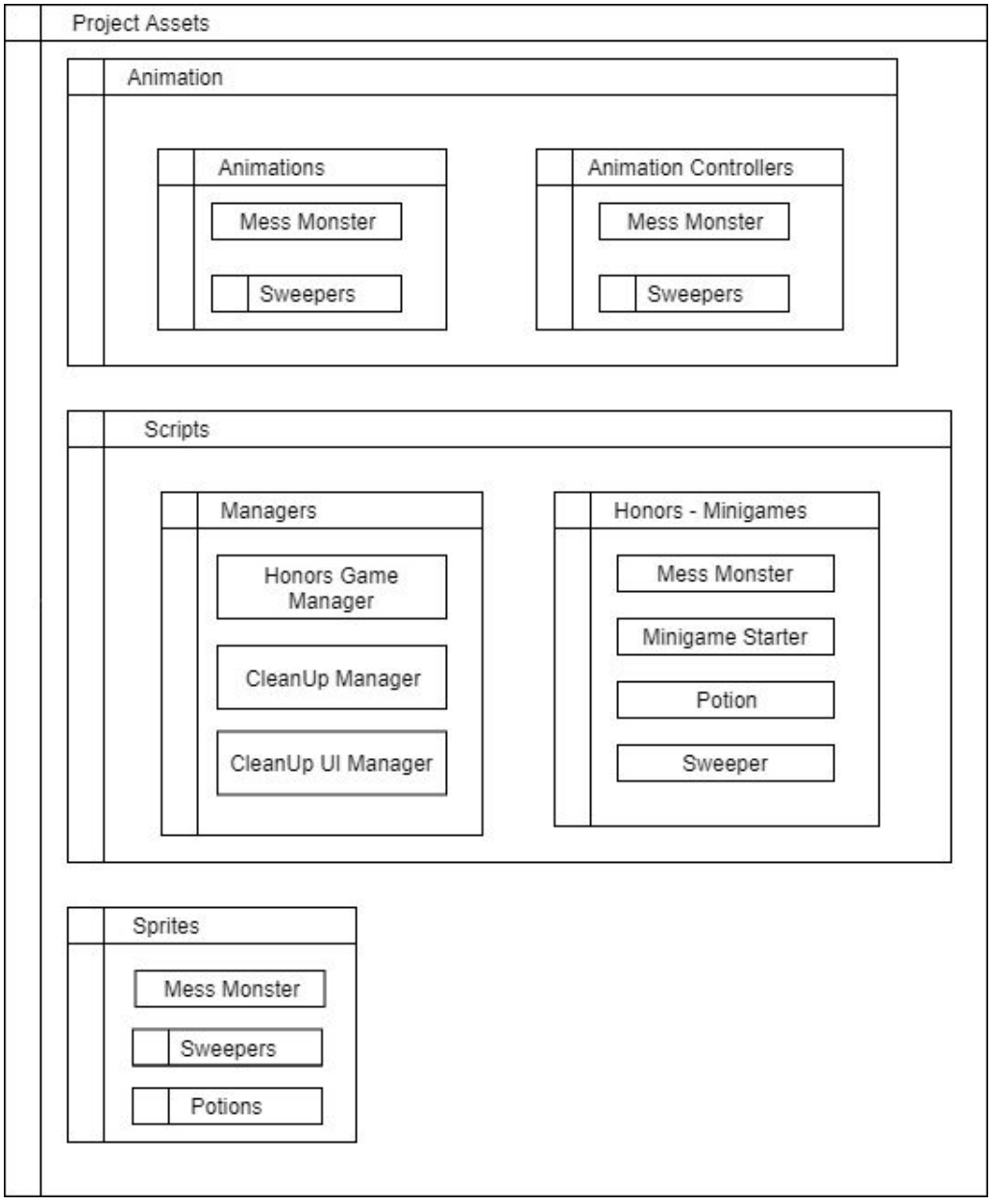

Figure 4.1 - Files written for the capstone

4.1.1 Animation

This folder contains the animations used in the CleanUp minigame as well as the animation controllers I made to control the animations in other scripts. These animation controllers are created and modified using Unity's built in UI, which can be read about in the documentation (Unity Technologies, 2019). The controllers are called in the scripting code and use state machines to determine which animation to play and whether or not the transitions between animations are quick or slow. The animations were made using the additional sprites found in the sprites folder. The animations add additional visual detail to the project that enhance the experience. Since the sweepers' animation are identical except for the actual sprites used, I was able to use an animation controller override, which substitutes sprites into another animation controller to create all but one of the sweepers' animations.

4.1.2 Sprites

The sprites found in the sprites folder are used either in conjunction with the contents of the animation folder or by themselves to give the game objects in Unity a visual appearance that can be used by players to quickly identify the object. The sprite sheets themselves were created by Bréana Townsend, one of my Capstone group members to ensure that the new sprites would match the style and quality of the rest of Grant Anon. I brought them into the editor and processed them for use in Unity. Since the sprites were given in sprite sheet form to easily organize types of objects and their animations together, I had to cut up the sprites with Unity and organize them into animations for later use. The Sprite Editor tool I used for this is well

documented in Unity's official documentation (Unity Technologies, 2019). The sweeper and potion sprite sheets contained multiple color variations of the same sprite, to distinguish between players and to provide an interesting array of potions for the monster to throw.

4.1.3 Scripts

Scripts contain the bulk of the actual coding work that went into the project. These C# scripts control the behavior of the project and allow me to customize the behavior of the Unity Engine to create my minigames. The scripts created for this project fall into either the Managers folder or the Honors - Minigames folder. The function of each file is listed below:

- **Honors Game Manager** - A game manager built off of the MainGameManager script used by Grant Anon, the Honors Game Manager contains additional methods to manage the starting and stopping of minigames within the usual gameplay loop. These methods pause and play the usual game objects found in Grant Anon while the minigame is being played.
- **Clean Up Game Manager** - A game manager built to control the gameplay loop while the minigame is being run. It contains scripts to begin the minigame, run the minigame, and end the minigame. It contains a pointer to a Clean Up Game UI Manager to load the start and end screens for the minigame.
- **Clean Up Game UI Manager** - This script handles the UI elements required by the minigames. It loads the start and end screens and listens for user input while the UI

elements are active. If input is detected, it calls the Clean Up Game Manager to respond appropriately.

- **Mess Monster** - Script that controls the mess monster found in the middle of the clean up game. It maintains a list of potions that is randomly throws around it in a certain radius to a randomly selected target destination.
- **Potion** - Potion are objects thrown by the mess monster, which call their animation script when they reach their destination to break. The Potion contains the actual mathematical calculations for traveling in an arc between its origin and its destination.
- **Minigame Starter** - Contains a timer for how long it waits between minigames to start another one. It also has a collider script to detect nearby units. If the timer has run out and there is a unit within its collider radius, it begins the minigame.
- **Sweeper** - The sweeper script listens for its player's input and moves around accordingly. If it makes contact with a potion that has reached its destination, it will destroy the potion and increment its player's score.

4.2 Final Product

The final version of the project can be found at the GitHub repository found at https://github.com/JustinRobbins7/GrantAnon/tree/Justin-Honors-Dev, the Justin-Honors-Dev branch of the main repository. A downloadable file of the project can be found here: https://github.com/JustinRobbins7/GrantAnon/releases, which contains the source code and a zip with the built project that can be run to play the game.

5 References

Partin, W. (2018, July 13). *StarCraft II': How Blizzard Brought the King of Esports Back From the Dead.* Retrieved from https://variety.com/2018/gaming/features/starcraft-ii-esports-history-1202873246/

Super Happy Games. (2017, May 8). *What is a Mini Game?* Retrieved from http://bestminigame.com/what-is-a-mini-game

Unity Technologies. (2019). *Unity User Manual.* Retrieved from https://docs.unity3d.com/Manual/index.html

Unity Technologies. (2019). *The Animator Controller Asset.* Retrieved from https://docs.unity3d.com/Manual/Animator.html

Unity Technologies. (2019). *Sprite Editor.* Retrieved from https://docs.unity3d.com/Manual/SpriteEditor.html

Wayward. (2015, September 25). *What is a Real Time Strategy Game? An Exploration and Definition.* Retrieved from https://waywardstrategy.com/2015/09/25/what-is-an-rts-game/