GIMM Group Project Rhetorical Analysis

For this project, we were thrown into the world of AR head-first. AR is something I was sort of familiar with, growing up I had a DS and that introduced little cards that you could scan that would bring up little figures using the DS Camera. That was probably my first introduction into AR. After that, I had played Pokémon Go when it was popular which put Pokémon in real world spaces. So, I wasn’t unfamiliar with the aspect of AR. However, once I took a step onto the other side of the screen, it was a lot different. There were many different programs that you could use with Unity to get AR to work. Our group started with AR Kit. When we started, we gave each other roles due to the project instructions saying it would be better to split it that way. We also made sure to evenly distribute the work to each other, so everyone had a task to do and a completion time. I took on the role of the programmer because I felt like and still feel like I have a good understanding of programming and C# and also how to use it in Unity. My first goal was to get the State Machine working for the game, so I really didn’t dabble at all in the AR field. That job was given to Wayne and William, my other two group members.

Our idea for the game seemed like it could be easy in our minds but was actually fairly complex. When we were introduced to AR “Markers”, we took the word marker seriously. Our idea was that you could draw with markers on a whiteboard to make the game objects spawn into the game field, and then someone could play the level on their phone. It would be a two-player game where one person challenged the other. One person drew the level and tried to make it challenging with a few bounds, and the other person tried to complete the level. It would go back and forth until either someone reached a score limit, or someone ran out of lives. A good game idea. We decided to take the graphics and code and objects from William’s individual game, because we believed it would work the best with this concept.

So, Wayne and William were tasked to get the marker recognition to work with phones. Unfortunately, after a few weeks, they had made no progress on this while I had finished the State Machine. There were a lot of weird things you had to do and download for Unity to broadcast to your phone. It took us many tries but never ended up working. We switched to Wayne’s laptop which had a webcam on it and noticed that the markers were picking up. After I started doing my own research, I decided that we should switch to Vuforia because it is what was recommended to use. Once switching to Vuforia, we were able to make our own markers and download them to Unity. These markers were too simple however, Vuforia needs complex makers with lots of points to identify what the object is to be able to scan it. William joked about using the Sprite logo as a marker, but in my head, that didn’t sound so bad. The Sprite maker worked wonders and it was able to spawn something using AR through it. The first real progress we made on the AR portion.

However, this was in the week before the final week. We had almost nothing to show because it had taken us so long to realize that 1. AR Kit isn’t working and 2. The idea of drawing markers on whiteboard isn’t going to work either. We had to shift our idea. The new idea was to take these markers and place them around in the real world and that they would get placed in the game field somewhere close to where it is in the real world to build a level. We got all of our game objects and makers working, and I was able to develop a script that spawns a prefab of our game object when the marker is scanned. When a marker is scanned, an object just appears perfectly mimicking where the marker is in the real world. When you stop scanning, that object disappears. A big hurdle was discovering how to keep this object on the screen so that it can be used to play the game. At this point in time, we also had to ditch the idea of using a phone because we had no success with Vuforia picking up phones as a device either. Luckily, I have a webcam that we were able to use, and it scans the markers perfectly. In Vuforia in Unity, the markers have commands called “onTargetFound()”. I knew this would be perfect, it was just like the command for when a button is clicked. All I had to do was throw in code of spawning a prefab and it would be able to spawn it into the game world. And it did. But, once messing around with this, I learned that scanning the markers really only spawned the game objects in one general area in the game field. Instead of trying to work it out to no avail, I learned from my mistakes and immediately pivoted. I decided to implement four more markers: Up, Down, Left, and Right. Once you scanned an object marker, you would then scan a location marker. I made one script called “SpawnManager” to handle all of this. It is around 500 lines, but I would like to believe it is organized nicely. When an object is spawned, it assigns a string value to a string “Object” telling it what object it is. For example, our Dr. Pepper marker is attached to the platform object, so it would assign the “Object” string the value “HorizPlatform”, as there is the horizontal platforms and the vertical platforms, which are spawned by the Coke marker. The Mug marker spawns a spinner, which randomly selects what direction it will spin in the “SpinnerObject” script. The Sprite maker spawns a turret, all of the turrets spawn on the walls of the game field and are rotated the right position thanks to the “SpawnManager” script. The spawn manager continuously runs four methods in the update section. These methods constantly check what the string values are. If the string values ever read “Spinner” and “Right”, it will spawn a right spinner from a list of random locations of spinner that are chosen by a random number generator. This is true for every object. They all have different spawns according to what direction and object you chose. That is why this script has about 500 lines. I added comments so it is easier to read and navigate. That is the biggest script in the game.

One script known as the “Camera Freeze Script” made by William disables the AR Camera and spawns a regular camera when the Enter key is pressed which allows you to transition from build to play mode. There is one track of music that plays in build mode and another track that plays in play mode that were made by Wayne. The game field is just skinny cubes to make up a barrier and a purple background. Every object gets spawned inside of this field. The player and the end goal randomly choses a platform to spawn on. This is shown by the “PlayerSpawn” script and the “EndSpawn” script. The player just has to collect the coin to complete the level. Then they get to build the level for the other player. The scene simply resets when either a player dies or wins so that the AR camera is reenabled and all of the game object prefabs are destroyed. However, the Keep UI script is on the score and lives, so they do not destroy on load. The concept of the game changed a lot, but we were able to keep the player vs player feeling, and I believe it is fun in its own way. The level creation can be a little slow because you have to scan many things to spawn in the objects, and the player can beat the level very quick, most of the time playing will be scanning markers. But I believe it is a unique AR experience.

As far as rating my group members go, I would have to give William a 4 and Wayne a 2. After our evaluation and the realization of scoping ourselves and the closing deadline, I took the project by the reigns which I had failed to do before the evaluation because I believed that Wayne and William would be able to complete their tasks. I almost completely made the markers, and the game objects work and spawn inside of Unity as well as writing all of the code. William was at least willing to help and made a few graphics such as the player lives and made the camera freeze code to help. Wayne however, spent most of his time working on the music tracks. From the beginning he wanted to make music tracks, but I kept telling him that he should save those for the end after his real task was done. They failed to pivot during the first two starting weeks, and we had to pivot after the evaluation week. I can’t say that almost building the entire game in a week wasn’t fun. It was really satisfying writing all of the code and putting everything together watching as it worked. Basically, everything for this project after the evaluation was just me theoretically thinking in terms of coding and Unity. Everything I thought would work did work, and I was very happy when all of the hard work was put into one project. Hopefully in the future when another group project comes up, I will have better communication with the group to know if they are struggling and if they are on track so I can help when necessary, instead of doing a final week push.