**CGS Contribution**

**Rift Jumpers(Alpha)**

Started with creating the dimension shifter script which controlled the current dimension and the progress of that dimension. As if the player was in that dimension too long the game would end. To allow for this to be used elsewhere in the project I created unity events for these transitions so that they can be called whenever needed.

Next, I started work on making sure that the wall destruction hooked up with the shield bash. To do this the script needed to be reworked to allow for this to all be hooked up properly. I created the system that allows for the destruction to happen when the player right clicks next to a breakable wall.

I created the players health bar and created the enemy health bar and hooked that up.

The paladin knockback was next which was a big system as we wanted the paladins shield bash to push enemies back. This was originally a big problem as we were using nav mesh agents therefore we can’t use a rigid body or much to apply a force. So, I used a rigid body and toggled the nav mesh agent on and off for when I wanted to push the enemy back. All this came down to then was balancing the numbers to make the push back look and feel good. If this was too strong, then sometimes the enemies could fly into the air and lose the floor which wasn’t great but fixing this was just playing around with the drag and angular drag of the enemies.

I also helped with testing the game and trying to find any major bugs or flaws in any systems.

As well as this I documented these systems on the git wiki.

**VR Game(Beta)**

At the start I was working on networking, so I first did research into networking vr games where I found that Photon Pun 2 was seemingly a good and easy way to network games for unity including vr. So, then I tried to set up a scene where two people could see each other in vr. Once this was done, I started to work on networking animations. I then started to create a lobby system where players would be able to join each other’s games through a code.

This led me to create a keypad for vr as all the ones on the asset store i couldn’t get to work. This keypad allows the user to input a code with letters or numbers. This was used to allow the user to input the room code and then join the appropriate session.

Next, I created a base32 code system that uses 32 characters to generate a six digit code this was so the room codes were randomly generated and wouldn’t be the same each time. As well as this it helped with ensuring the games code was secure.

Next I shifted focus to the actual game loop which was working on the 3v3 arena manager script which is the script that deals with the game itself and manages the timers of the game and the different states. This also meant working to ensure that all of these features for the game loop would be networked and the same on every player’s game. This was done through using photons RPC calls which allows me to call a function on a specific group of people in the game. This was useful to make sure that timers were always synched up together and making sure that every player in the lobby knew about everything they needed to at the same time. For example, of the teams score was fired across using an RPC to ensure that every player had the correct score for their team.

Then I started to work on focusing on getting player spawns hooked up for the game and the post match area so that the game loop would be finished and could then be improved upon with effects. To do this I created a function that would spawn everyone on that team on a point using targets in the scene. This would increment through as players joined with the host being the one to set this up for joining players.

Next was finishing up the game loop to ensure that players could play more than one game through and that the game would allow for this. So, I started by checking if any in game values needed to be reset on the arena manager as this was something that could break the game another run through. Most of this was all good to go with how the arena manager was set up to begin with as the scene was reset between loading into a new game however the spawn points counters did need to be reset to make sure that players spawned in the correct spot. This know meant that we had a working game loop.

As well as this me and conrad investigated including host migration into the game. This was because if the host dropped from the game, then the game just fell apart as the RPCs were no longer being sent out. We realised that we would just need to ensure that who ever got the new host would be at the same point in the game as the old host was so that the game was fair and could still be played out. In the end we didn’t have time to fully implement it however this is written down in git issues and it should be a couple of variables that need to be checked.

Next, I did some small work on implementing some visual effects some of the objects for example when a player breaks a wall using the heavy class.

I helped play test the game and make sure that everything was running correctly with playing the game through and the classes.

All these systems have been documented on the wiki as well.

**Autarky(Gold)**

I created the base system for the workstations which took a metal and would spit out a car. This system was extremely complex and ended up needing to be reworked as it didn’t end up working well. I created a manager that would spawn a workstation line which is a spawn five workstations and an output. I then used gizmos to show the developer the direction of the workstation line. I then created a system that dynamically created buffer points at runtime so that when the metal was supposed to move along the line it had places to go and stop. This system would also work no matter which direction the workstation line was facing. Then I started to get the workstation to spawn metal and started to work on getting that metal to move. However, the way that I was doing wasn’t the best as it involved a lot of parenting to try to keep track of which workstation the metal was heading towards. This is the reason why I then handed the system off to be reworked.

I started to create the breakroom for the pc client. This room needed to allow workers to path into it and then the manager to also be able to walk in. So, I created a door that opened and closed when the player went near it. Once the manager was inside a button would appear on the players ui that would give them the option to “yell” this action would remove any workers that were currently in the breakroom and send them off of break back to work.

I created the factory manager. This script deals with everything in the factory and is a singleton that can hold references to objects in the scene. This then lets everyone get references to those objects easily without the code being too messy. This isn’t the best way to do this however its an easy addition that allows references to be got easy through getters and setters if necessary. This includes the factory money which this script controls and has functions that adds and takes away money from the player based on actions like adding metal etc.

As well as the break room I added some code in the pc client that would mirror the webgl build of the game for newspapers and articles. Which would give workers happiness depending on what article they would read. However, after working on this system, I realised that instead of mirroring the system it can be done by just sending over values to change the worker happiness which is a lot easier and simpler.

Next, I worked on creating manager actions this is actions like the yell. I created a speech wheel that allowed the manager to shout at the works and give them instructions on how fast to work and what quality car to produce.

On top of this I added worker UI to the pc client that shows an outline when the manager hovers over the workers this is so that the player can easily see which worker they will select if they were to click. This also includes a happiness bar above the worker which has different colours based on how happy they are and the workers name as well.

**Solo Project**

I wanted to recreate the portal gun from portal in the new unreal 5 engine as I’ve not really got to use unreal engine before so I thought that it would be good to try to learn it as it’s a big part of the games industry. I wanted to create a portal system as I knew this system was quite complex and hard to do well. I started by trying to get a simple teleport working with some nice effects. Once this was done I tried to make it more dynamic by only allowing portals to be placed in places that they actually fit on the wall. Next I tried to get the teleport transition to be smoother. This was done by adding a delay on the teleport and playing around with the physics checks to allow the player to walk through the wall and then teleport the player further through the target portal. Once the portal system was completed, I started using unreal 5s features to create a level that could show off this mechanic. Then I added some assets and some sounds.