Team 2-bit Dev Log

Week 6: 23/10/23 - 27/10/23

Michael Hayes

Overview:

This week we finally resolved the input issue plaguing us for the past 3 weeks, although unfortunately we have run into yet another blocker. As we have low experience with VR, and a strict timeline, we discussed moving the project to the local multiplayer brief. Our final decision will be made next week after thinking over it.

Agile Sprint Update:

Sprint 4: Prototype (In-progress)

Production of 2 game proposals with detailed information on the experience. To be submitted to Liminal VR for green light to begin development.

Links

Link to Minutes and Agenda: CS2 - Minutes and Agenda
Link to Team's Jira Scrum board:
https://cs2mr.atlassian.net/jira/software/projects/CS2LVR/boards/2/backlog
Miro board overview: https://miro.com/app/board/uXjVMj-Nye0=/?share_link_id=77318451600

Table of Contents:

This week's completed tasks:

Week 6: 23/10/2023 - 27/10/2023

Monday:
- Business lesson

Tuesday:
- Finished the shooting mechanic, Laser guns can now only destroy matching colour targets

Wednesday:
- Troubleshooting why the inputs are not working

Thursday:
- Received comments from Liminal on Inputs not working - When Liminal build the project the inputs are working
- Miguel created new project with working inputs and shared it
- Tested the game to check if the inputs were working and forked the project
- Troubleshooting on the raycast end position not updating to controllers rotation

Friday:
- Exported and imported all project files to the newly forked project
- More troubleshooting on the raycast end position not updating to controllers rotation
- Moodboard and local multiplayer tank game design

Sprint 4 Tasks:

Hitting the correct target mechanic

The main mechanic of the game, being performed 100s of times a session, is to shoot the correct colour target with the matching colour target. Only when the correct pair meet, the target will be destroyed.

To make sure the game is as optimised we first thought about the best way to perform the laser shooting a target mechanic in code. There were two main ways of doing this:

1. Have the laser gun listen to a target before performing a function.
2. Have the target listen to the laser before performing a function.

We decided that having the target listen to the laser gun before performing a destroy function would be a better idea as there will be multiple targets within the scene, and only 2 guns.

First I made the code for checking which target was hit. To do this I needed a reference. I choose to use a class as a substitute for a tag as tags are generally more difficult to debug and they also deterred in the Liminal SDK according to their wiki.

There will be no start or update loop within the tag so this should hopefully not cost too much performance. These classes will be attached to a target prefab variant, to define which target colour they are. The targets will also be using an object pool so there will not be multiple targets being instantiated and destroyed.

Target Class:

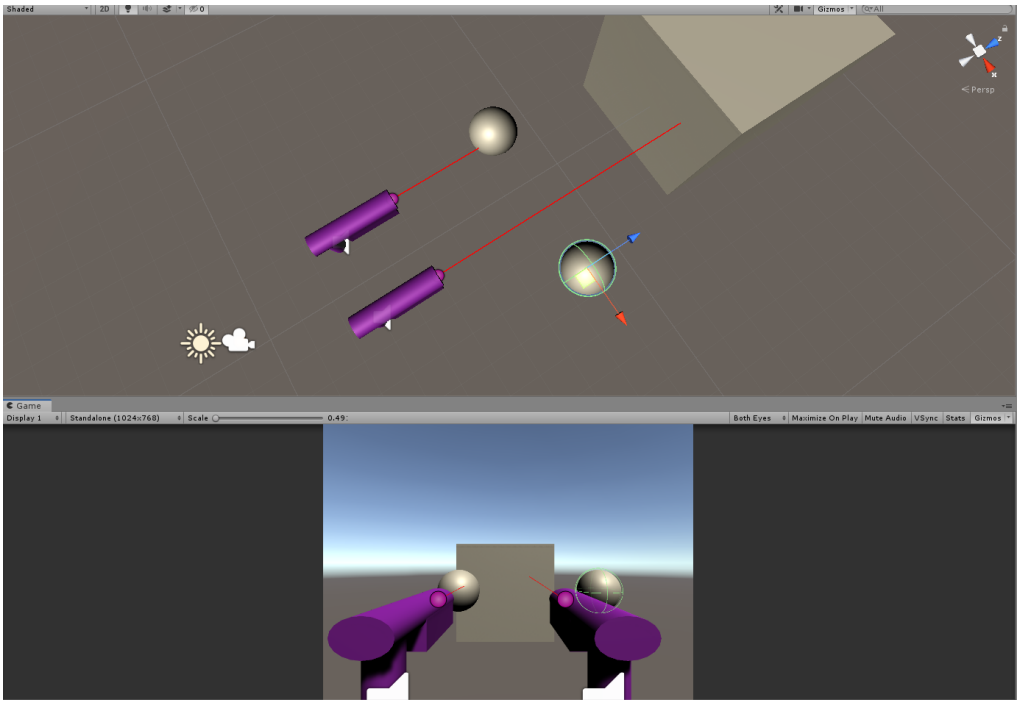
```
2
      Unity Script | 1 reference
3     public class Target1Tag : MonoBehaviour
4     {
5         // This class is used as a Tag to reference the attached game object
6     }
7
```

Code referencing the target class:

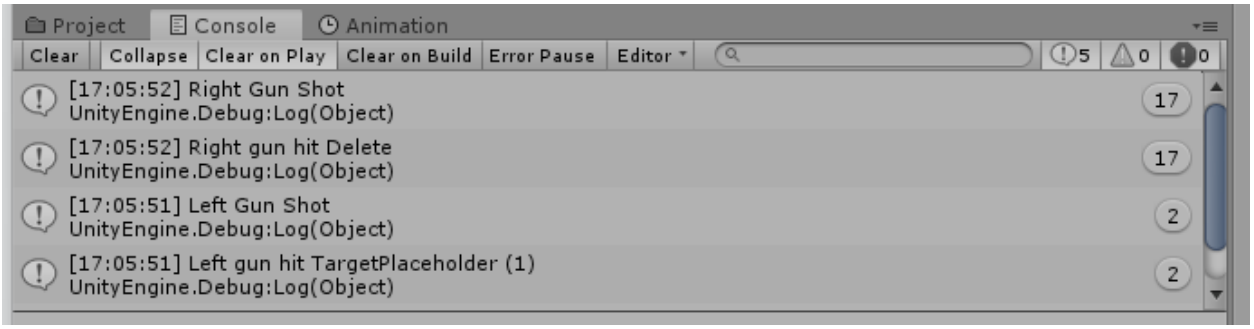
```
1 reference
private void CastRayLeftGun()
{
    RaycastHit hit;
    Ray gunRay = new Ray(leftGunMuzzle.position, Vector3.forward);

    if (Physics.Raycast(gunRay, out hit))
    {
        if (hit.collider != null &&
            hit.collider.gameObject.GetComponent<Target1Tag>())
        {
            lastRayHitLeft = hit.point;
            Debug.Log("Left gun hit " + hit.collider.gameObject.name);
        }
    }
}
```

Targets in Unity play mode:

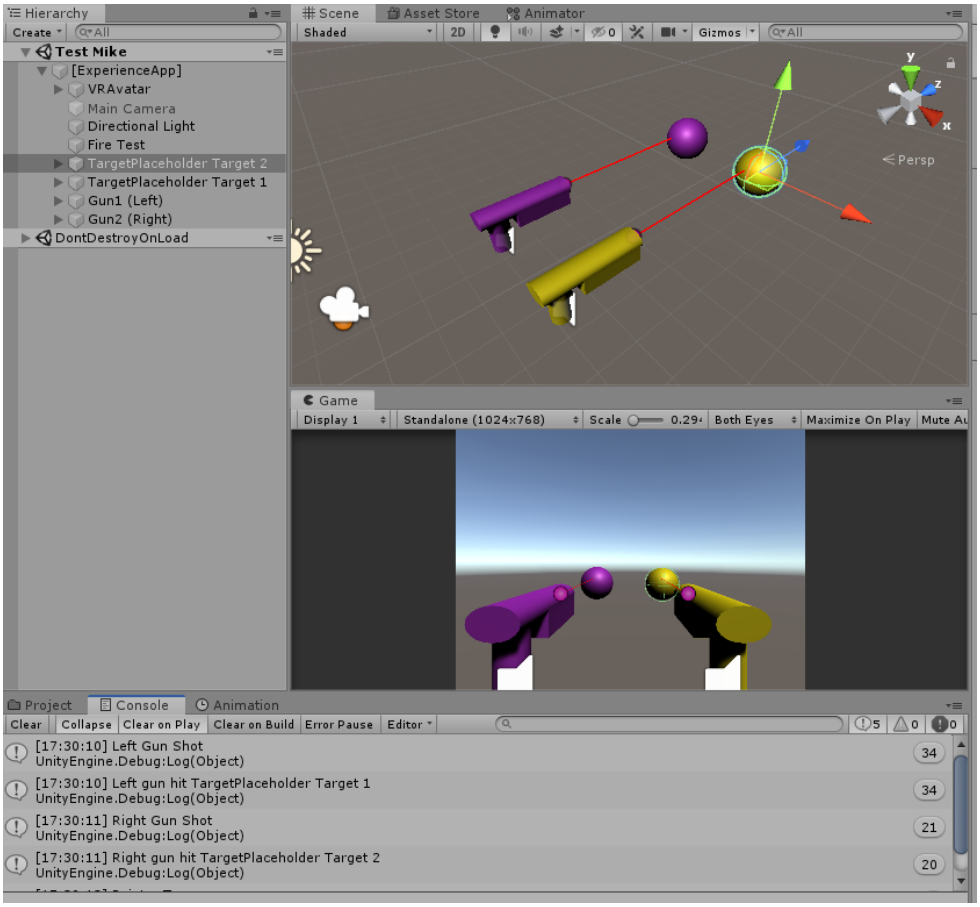


Console showing targets hitting correctly:

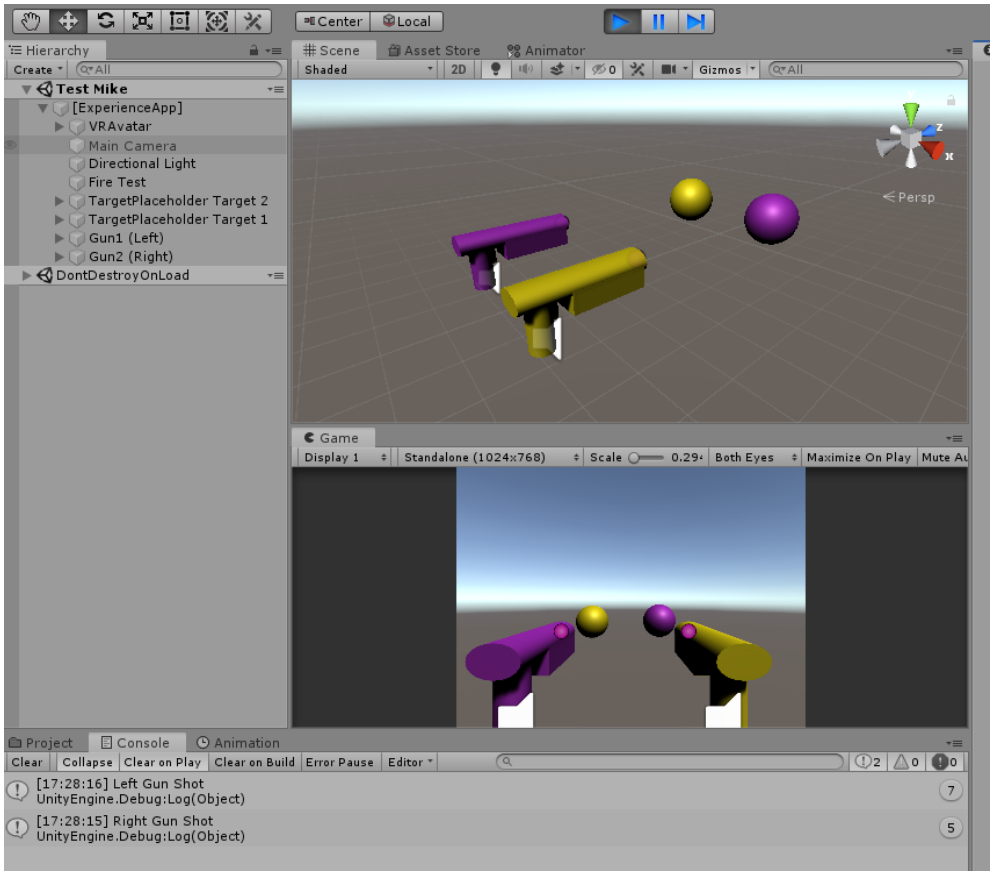


Creating Target Variant / Testing

Through this test, I found a bug. When shooting rapidly, the code checking the raycast hit may not be called. To Resolve this, a delay after each shot would be required.



When the correct guns is not used:



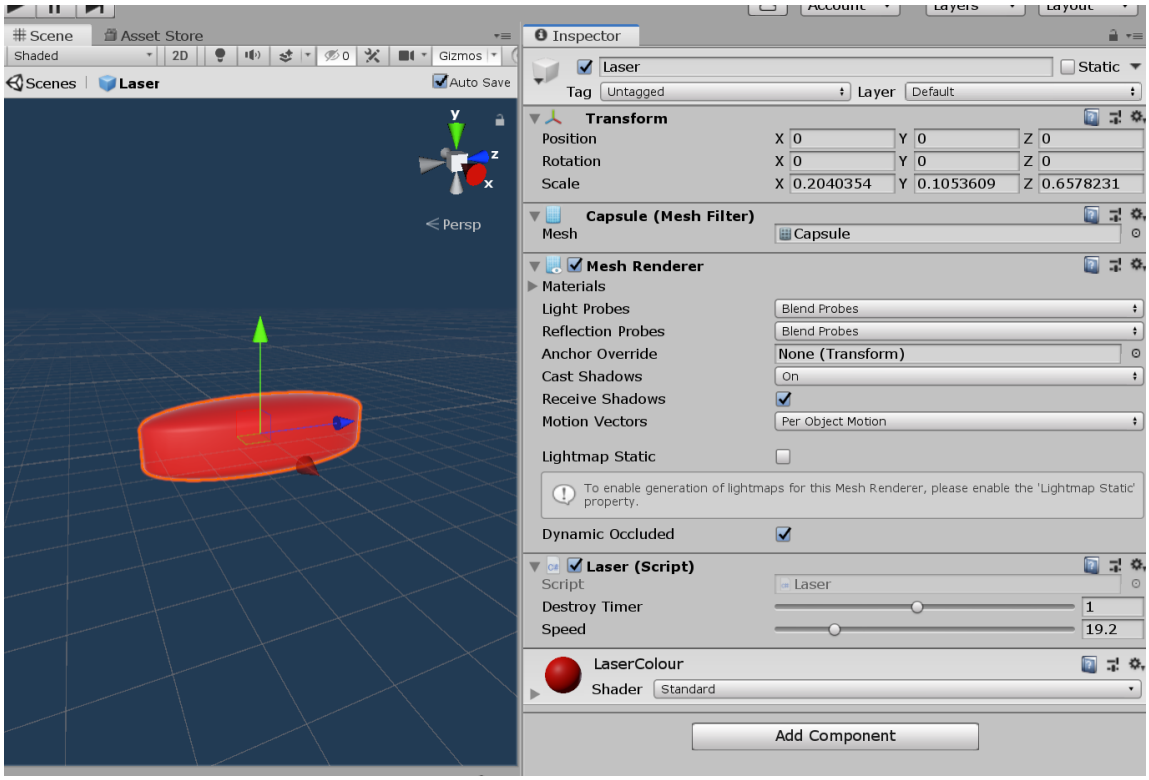
Creating new project files

After sending our project to Liminal VR due to our input issue not being solved, they came back saying it was working in an Android apk file. Unfortunately, I was unable to build to my oculus quest as I do not have a developer account and the unity project was not detecting the oculus quest.

Another classmate Jesus Miguel, had a project that would play in unity play mode through the steam vr app which did not work for us. Jesus kindly made a project and set it over for us to fork and use a base. This project worked with the Oculus headset in play mode through steam VR, finally allowing us to test, and the inputs worked as expected. I then exported the shooting and spawn system mechanics from the previous project and imported it into the new one. The same input code appeared to work in this project so it seems the project was set up incorrectly.

Creating laser visuals

As we want to create a well optimised project we need to think about everything we are implementing and how it could impact performance.
Therefore, I decided to go with a visual element only for the laser ray. As a placeholder, I created a game object, removed the collider and added a single script.



Laser Script

The laser script would be responsible for moving the laser game object attached to a set direction and then destroying it with the game object. To do this I used a coroutine to destroy the game object after a certain amount of seconds had passed.The position was set to its instantiated position and the would be updated to move forward continuously until destroyed. Ranges were used for the parameters to easily adjust values within the inspector.

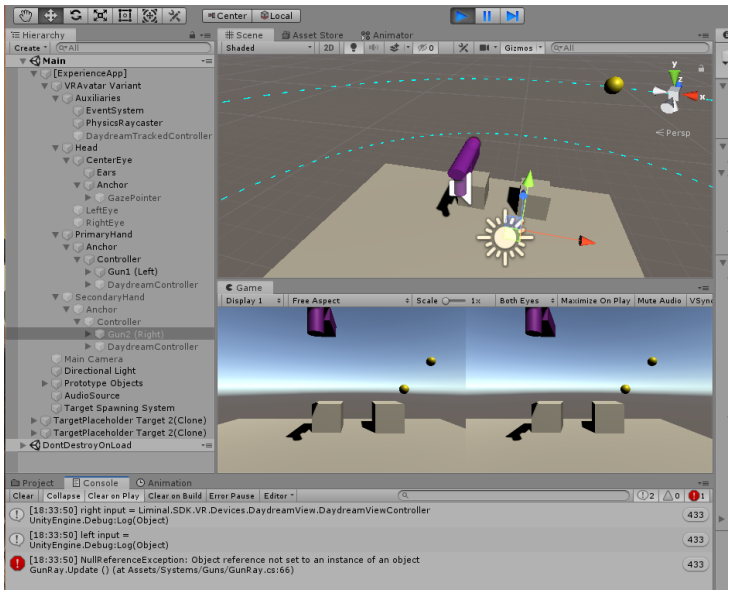
```
5   public class Laser : MonoBehaviour
6   {
7       [Range(0.1f, 2f)] [SerializeField] float destroyTimer = .5f;
8       [Range(0.1f, 100f)] [SerializeField] float speed = 5f;
9
10      private void Start()
11      {
12          StartCoroutine(Destroy());
13      }
14
15      private void Update()
16      {
17          transform.position = Vector3.MoveTowards(transform.position,
18              transform.position + Vector3.forward, speed * Time.deltaTime);
19      }
20
21      private IEnumerator Destroy()
22      {
23          while (true)
24          {
25              yield return new WaitForSeconds(destroyTimer);
26              Destroy(gameObject);
27          }
28      }
29  }
30
```

To refactor this code later, it may be more efficient to use a line render instead of 3d mesh with more vertices, or an object pool to reduce the amount of instantiating and destroying of game objects that occurs.

Challenges Encountered:

Null reference for left controller

Problem: When debugging the reason for our controllers not showing in play mode, we found through debugging that the right controller was producing a null error:



Solution: In emulator mode there is no controller for the right hand, we need to build the game in OVR mode.

Laser Guns not showing in Play mode

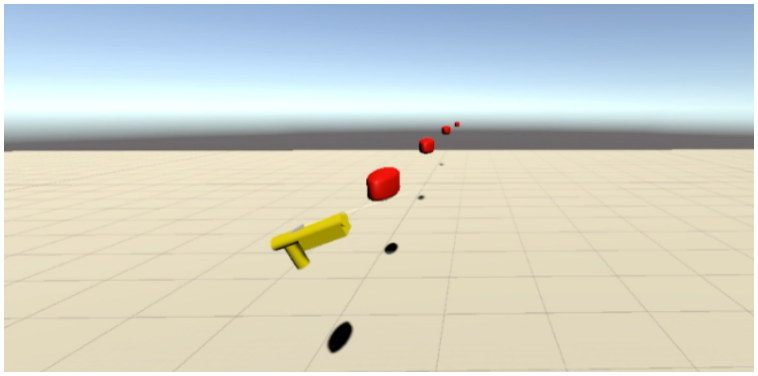
Problem: When testing the game in play mode in VR, we found the guns would disappear and not show up.

When steam is connected to the headset and the headset is on, the play mode setting in the experience app automatically switches back to Emulator mode, making the guns not appear.

Solution: To solve this we will need to build the game to test it. When the game is built, the controllers appear.

Lasers shooting in towards the same vector3 point (not matching gun rotation)

Problem when shooting the gun (using a raycast) the raycast and anything using the same logic (instantiated laser, gizmos) do not update their rotation to the gun / controller:



Code to shoot a laser using a raycast and Instantiating a mesh renderer

```
private void CastRayLeftGun()
{
    RaycastHit hit;
    Ray gunRay = new Ray(leftGunMuzzle.position, Vector3.forward);
    Debug.DrawRay(leftGunMuzzle.position, Vector3.forward, Color.red);
    if (Physics.Raycast(gunRay, out hit))
    {
        if (hit.collider != null &&
            hit.collider.gameObject.GetComponent<Target1Tag>())
        {
            lastRayHitLeft = hit.point;
            Debug.Log("Left gun hit " + hit.collider.gameObject.name);
            Destroy(hit.collider.gameObject, .1f);
            targetsHit++;
        }
    }
}
```

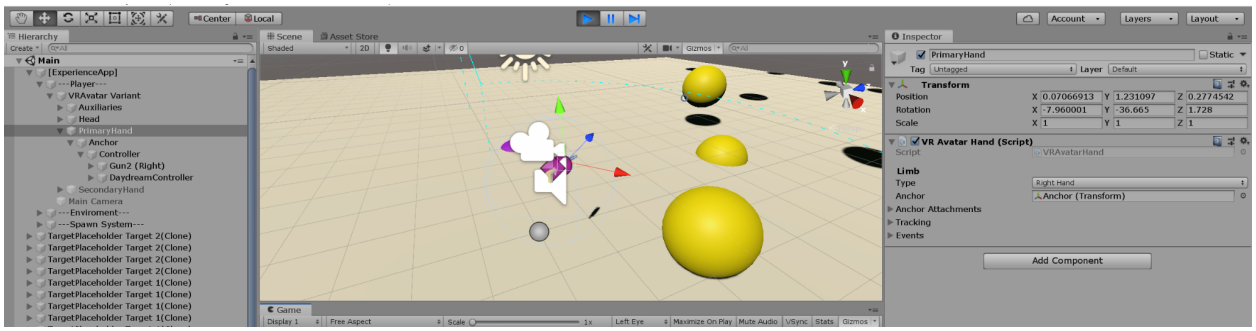
Code to make laser move forward from the gun

```
private void Update()
{
    transform.position = Vector3.MoveTowards(transform.position,
        transform.position + Vector3.forward, speed * Time.deltaTime);
}
```

With the help of Robin who already implemented a similar thing for his spawn system gizmos, we came up with the following code:

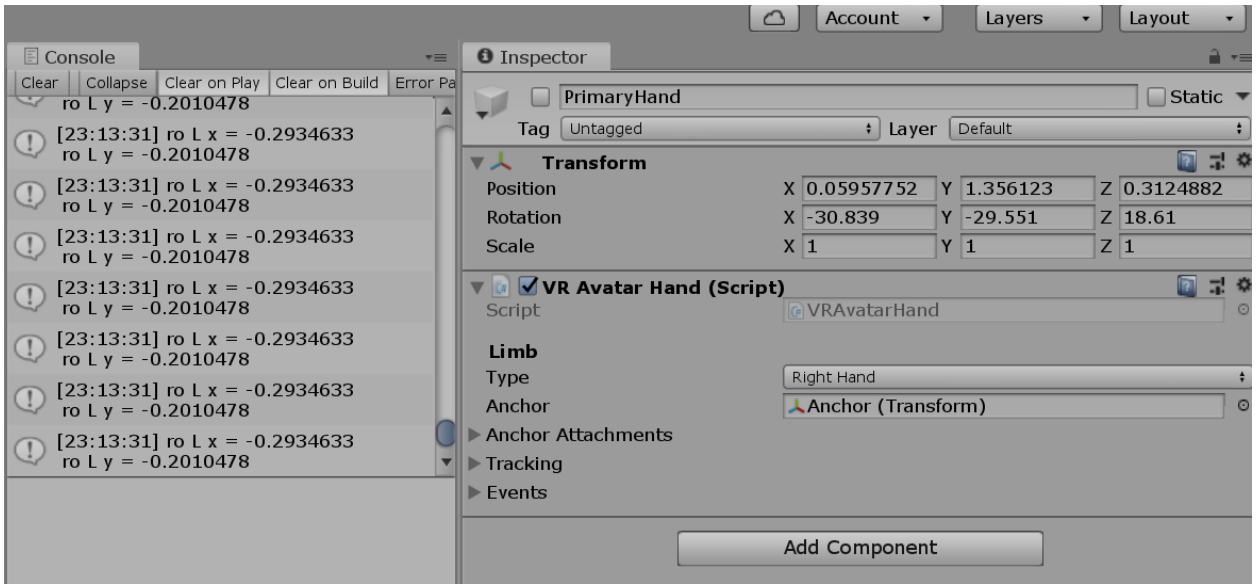
```
131     private void OnDrawGizmos()
132     {
133         Gizmos.color = Color.red;
134         float roLx = leftController.transform.rotation.x;
135         float roLy = leftController.transform.rotation.y;
136         float roLz = leftController.transform.rotation.z;
137         // Debug.Log("ro L x = " + roLx + "\n ro L y = " + roLy);
138 
139         DrawLine(leftGunMuzzle.position, roLx, roLy, roLz, 100);
140     }
141 
142     private void DrawLine(Vector3 startPos, float x, float y, float z, float distance)
143     {
144         Gizmos.DrawLine(startPos,
145             startPos + Quaternion.Euler(x, y, z) * Vector3.forward * distance);
146     }
147 
```

Here we went through the hierarchy whilst in emulator mode, moving the controller and seeing which game objects rotation was being updated. We found that the primary and secondary hands were the only game objects being updated.



To try and solve this we used the transform rotation of the primary hand and passed in its x,y,z rotation data into the draw line. Using the euler angles function we added the hands current rotation to the direction (point b) of the line. However, the hands rotation was still not affecting the end point of the gizmo.
Using a debug to see what information was being passed in for the rotation, we noticed it did not match the current hand position and thought maybe the position was being changed in the controller script causing our code to behave differently than expected.

Debug showing different values that expected in the primary hands current transform rotation:



Possibly switching briefs

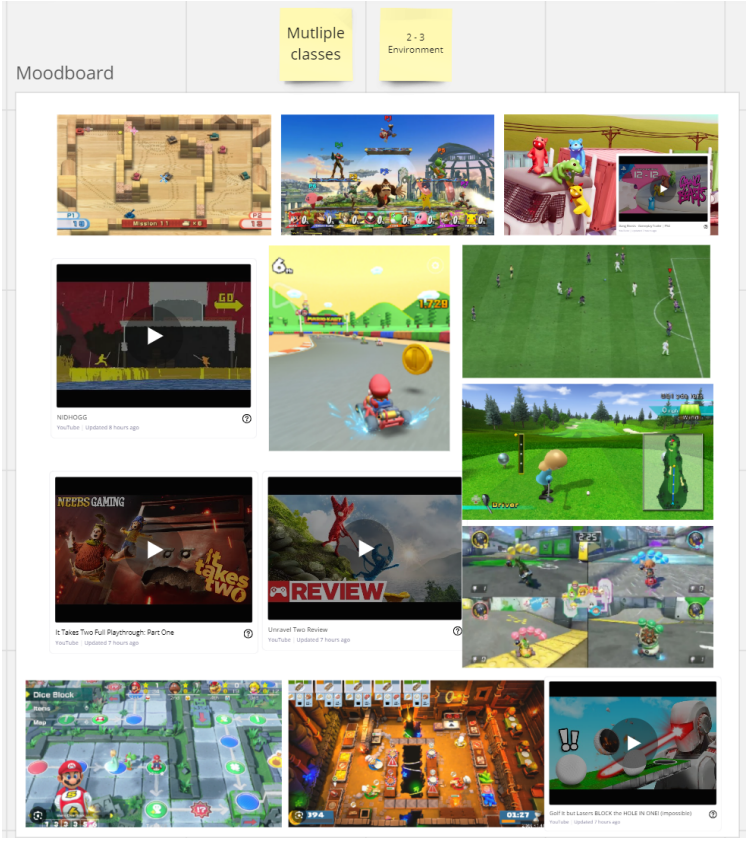
As we are constantly running into blockers with this project and unable to resolve them in a sufficient time, we are currently severely behind in the project. We both have lost a lot of motivation to continue the VR project as we are yet to have a successful week.

Therefore we got together in a meeting and considered the idea of switching project briefs to a local multiplayer game.

We felt that there are possibly many other potential blockers waiting for us. At this point, there are only 5 weeks remaining to finish the project. We are not entirely confident that we will be able to finish the project to a high enough standard. We do not plan to continue work on the project after the trimester ends, therefore, any hope of getting a published game on the Liminal platform is rather low.

Designing a local multiplayer game for a potential switch of briefs

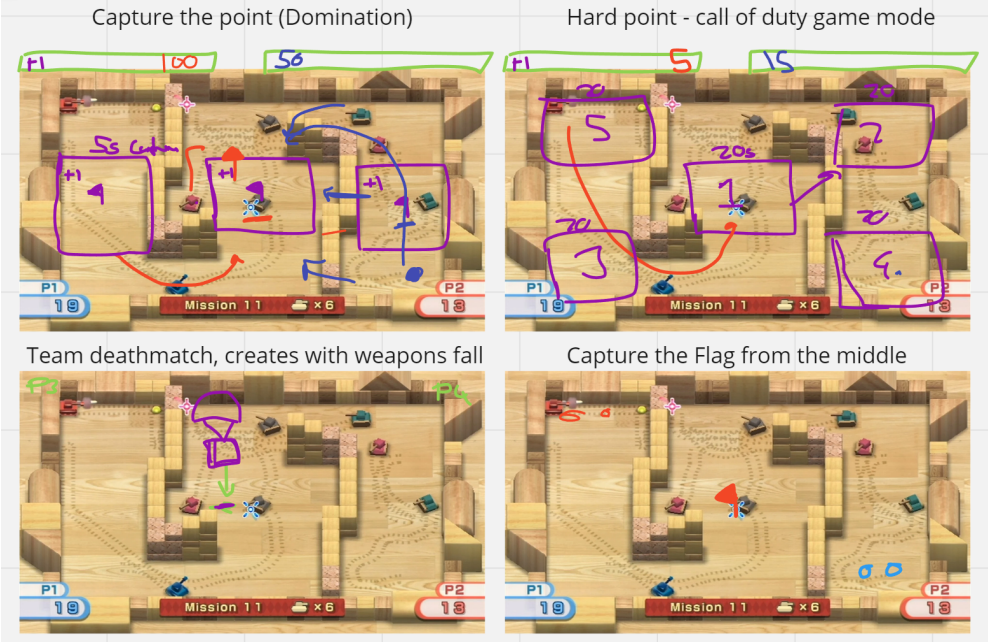
First we looked at some other multiplayer games and added them to a mood board.



We then came up with 4 ideas that we could use for our multiplayer game.

We then developed one of the ideas, the tank shooting game and thought through some 1v1 gameplay modes with the basic ideas: (using a Wii Tanks as a reference image for visual design)

- Players will control a tank vehicle
- Tanks will shoot rockets to damage other players
- The map will be 2.5D with a slight top down view



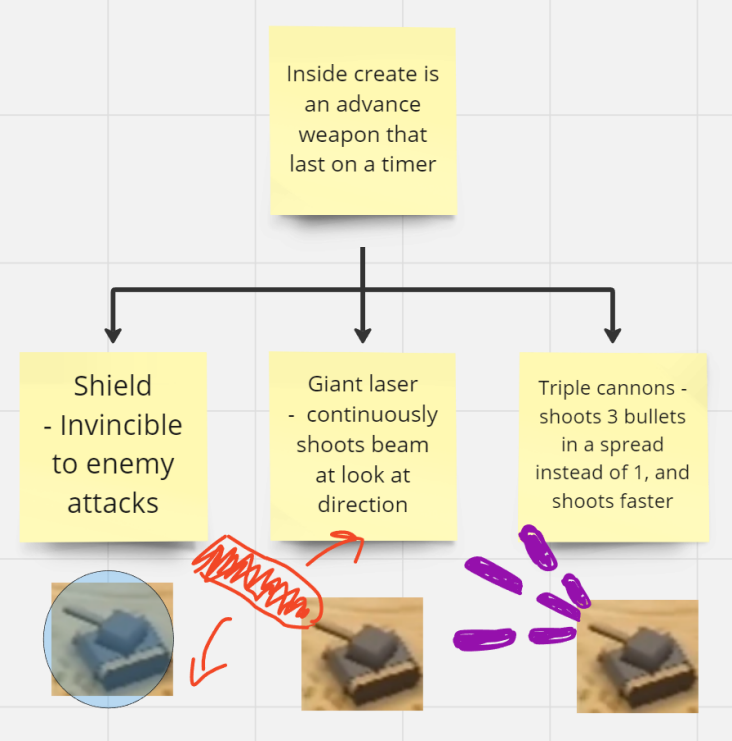
These all had multiple issues that needed to be considered to give both players a chance whilst rewarding skill and not relying completely on randomness.

We decided on a King of the hill gamemode where players would receive points only when inside the hill. This would remain statically in the middle of the map.


King of the Hill game mode
(1+ for each second in the hill)
Player 1 - Score
32
Player 2 - Score
87


Supply creates drop over time with powerful weapons
Player 1 - Score
32
Player 2 - Score
87

Inside the creates:



The idea of the advance weapons was to add an element of fun to the game and provide incentives for player to leave the hill, giving chances for the other to get back into the game:

Different classes and map themes will be selectable at the start screen.

Example of the different classes with personal stats:

Team Members:

Robin Pound - Co-Lead:

Individual works:

- Moodboard for possible local multiplayer game
- Started TDD for Laser Beat

Next Week's Goals:

- Complete sprint 4 (Prototype)

Sprint 4: Prototype

Production of 2 game proposals with detailed information on the experience. To be submitted to Liminal VR for green light to begin development.

Feedback and Comments: