VR Development Kickoff

Project Concept

During this session, we had discussions about what sort of project we would develop for the remaining time. We decided on creating a shooter-oriented game, where the player can perform the following actions:

1. **Pick up a gun.**
2. **Load a mag in the camber.**
3. **Kill some sort of enemy.**

After discussing the minimum actions needed to complete the project, we also talked about level design and potential visual improvements if time allowed. Our main goal was to prioritize having a functional game that fit within our time constraints.

Once the team agreed on the overall direction, we began working in the lab classroom, following the same collaborative approach and work ethic we used during our AR project.

We began by creating the project in **Unity Hub**, using a pre-existing template to set up our VR environment. While the template was being prepared, we created a new scene called **"Main."** Initially, we assumed we could simply import all the necessary assets into this scene once they were ready, using **Sketchfab** for models — but I'll cover that process later.

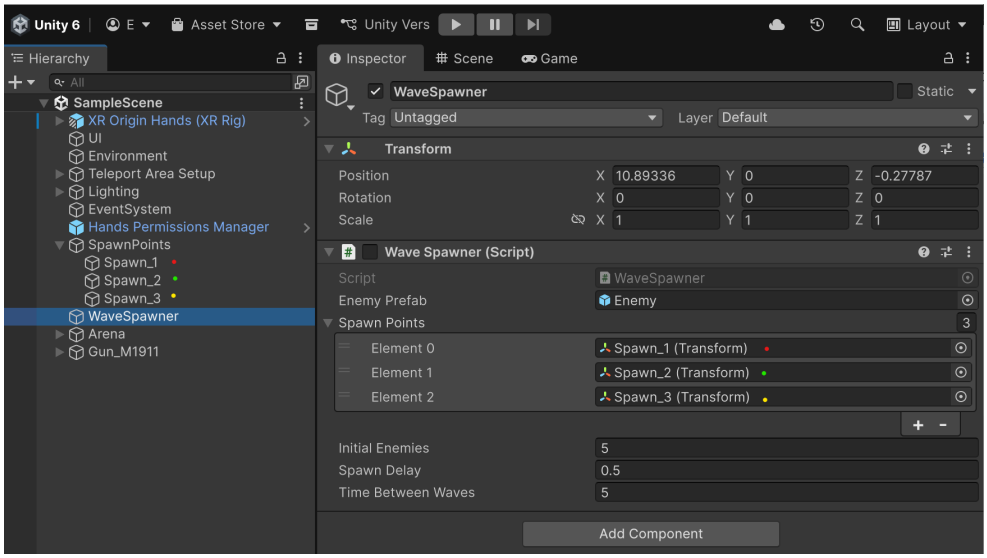
Next, we started scripting enemy behavior. The requirements for our enemies were as follows:

1. Enemies spawn on a set static location
2. Enemies will gravitate towards the player
3. Enemies have a way to damage the player, which if this action happens a lot activates step 4.
4. Enemies sends the player to a retry menu with text "Game Over"

Step 1 & 2 : Enemies & Movement

We created an empty GameObject named **SpawnPoints**, which contains three child objects: **Spawn_1**, **Spawn_2**, and **Spawn_3**. These child objects are referenced in a custom script called **WaveSpawner**, which handles enemy spawning. The **Enemy** script, attached to our enemy prefab, implements the logic for making enemies gravitate toward the player (step 2).

While testing, we encountered an issue where enemies chasing the player would clip, overlap, or fade into each other due to missing collision detection. To fix this, we added a **"Character Controller"** component to our enemy prefab, which resolved the issue.



Step 1 & 2

Step 1 : Picking up gun
Continuing, we went to **Sketchfab** to find a gun model to implement and chose a package called **“M1911 Pistol with magazine and bullet”** by DanaeH. We downloaded it in **.fbx** format and set up folders to organize the assets:

Base model location:
Assets → Art → Models → M1911

Base textures location:
Assets → Materials → M1911 → Textures

Texturing the base gun model took some time as we worked out how to properly apply the materials. Eventually, we located the model’s **Materials** tab and connected the necessary textures. There were three types of materials to layer: the **Base Map**, the **Metallic Map**, and others. However, adding the additional mappings worsened the model’s appearance, so we decided to stick with just the first two.

Next, we added a table and placed the gun and magazine on top. The gun was fading through the table, so we added **Box Colliders** to both objects, which resolved the issue. Once satisfied with the imported models, we began creating the logic to make the weapon interactable. We added the **XR Grab Interactable** component and a custom script called **GunController**. The GunController handles properties such as the muzzle (where bullets are fired from), projectile prefab, projectile speed, fire cooldown, whether a magazine is required (boolean), and magazine capacity. We also added **XR Grab Interactable** and **Box Collider** components to the magazine.

Conclusion: The player can grab the gun, and enemies spawn and approach the player.
Next Monday, we will continue working on the gun and implementing magazine loading.