Jekko Syquia
Into the Dungeon

VR Development Platform and Device

Unity Version

`Unity 2019.4.19f1`

Computer OS Version

`Windows 10 19042.746`

VR Headset Make and Model

`Oculus Quest`

Implementation Status:

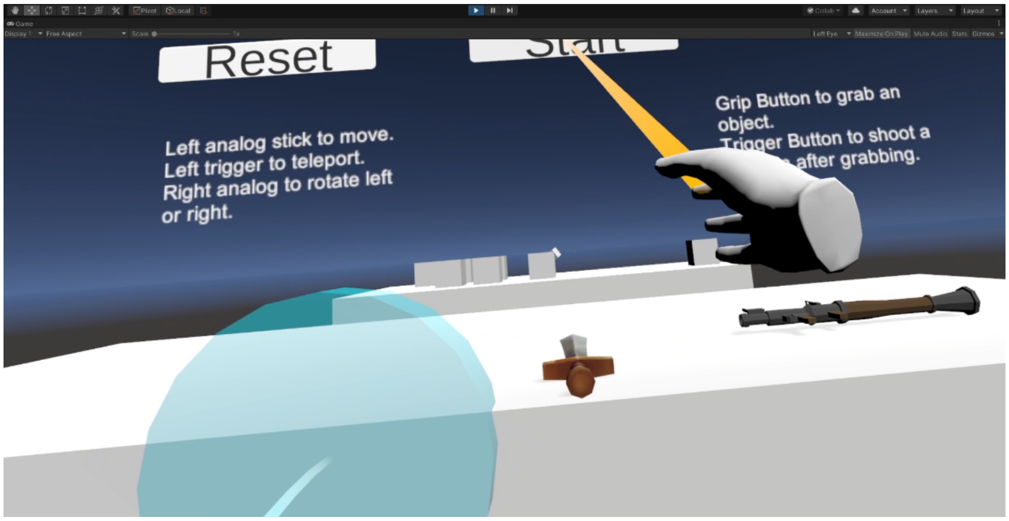
1. The VR design goals I was able to implement successfully are the following:
    - Player's head motions are in control the whole time.
        - The head motions are kept the same, therefore accurate to the users headset position.
    - Not overriding the field of view.
        - The field of view is kept the same, and did not add any effects on the wearer.
    - Avoided acceleration.
        - The speed is constant when moving to avoid motion sickness.
    - Screen space effects.
        - Similar with the field of view there are no edits made.
    - The frame rate was kept under 72 Hz.

Scene Switch

Below, I have created the option to move to the next room. This is still a work in progress I have also created the script SceneSwitcher to move to the next room.

```
public void switchScene(){
        SceneManager.LoadScene(sceneIndex);
    }
```

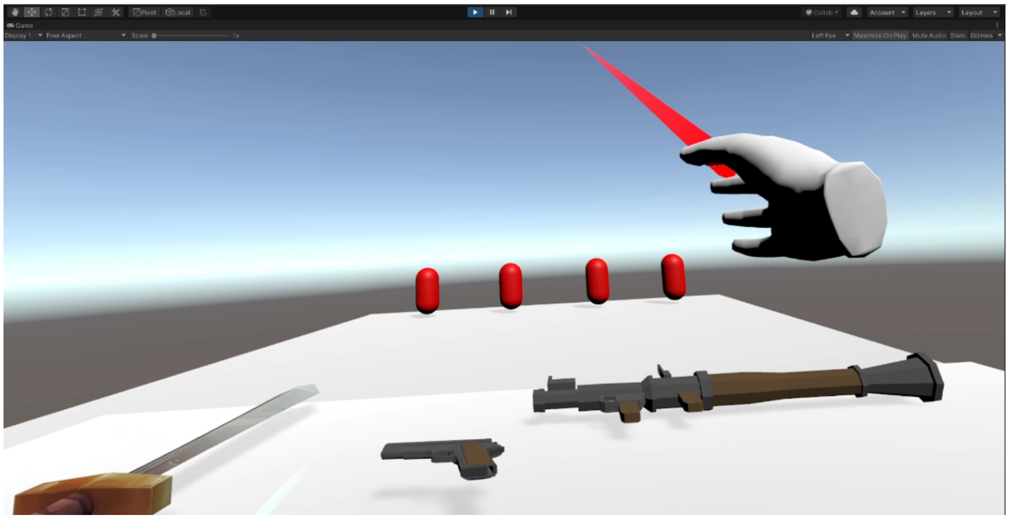
This script is invoked when the button `Start` is invoked. Demo shown below:



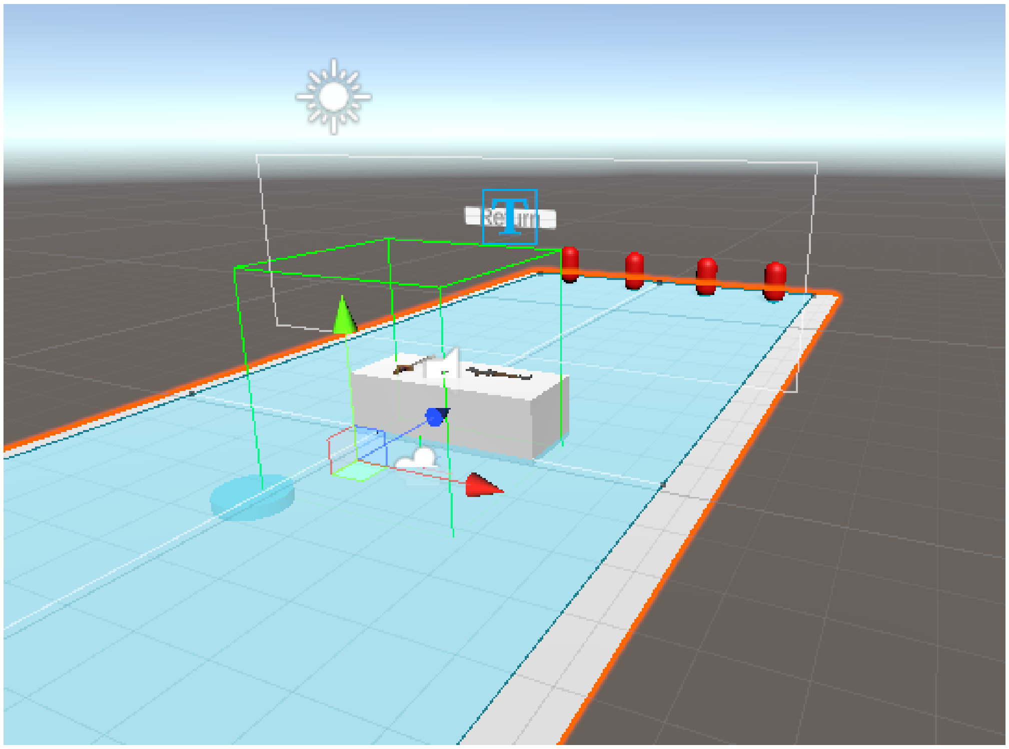
Left analog stick to move.
Left trigger to teleport.
Right analog to rotate left or right.

Grip Button to grab an object.
Trigger Button to shoot a
after grabbing.

*Demo showing scene transfer from **Tutorial Scene**.*



*Demo showing scene transfer to **Start Scene**.*



Start Map showing the layout and the red are the enemies. Blue indicates the region in which the NavMeshAgent types can walk in

Enemy AI

I have implemented my own script for creating an enemy AI that has 3 states. This uses the **Nav Mesh Agent** component to calculate the path to the player. The following states are:

1. Patrol
    - patrol the enemy moves randomly from the starting position.

    ```
     private void Patrolling()
    {
       if (!walkPointSet) GenerateWalkPoint();
       if (walkPointSet) agent.SetDestination(walkPoint);

       Vector3 distanceToWalkPoint = transform.position - walkPoint;

       // Reached the destination
       if (distanceToWalkPoint.magnitude < 1f)
           walkPointSet = false;
    }
    ```

    Above we can see the method `GenerateWalkPoint()` which calculates a random position for the enemy within the **x, z-axis.**
2. Chase
    - When a certain threshold is reached aka player is in **Sight Range**, the enemy starts chasing the player.

```
 private void ChasePlayer()
{
    agent.SetDestination(player.position);
}
```

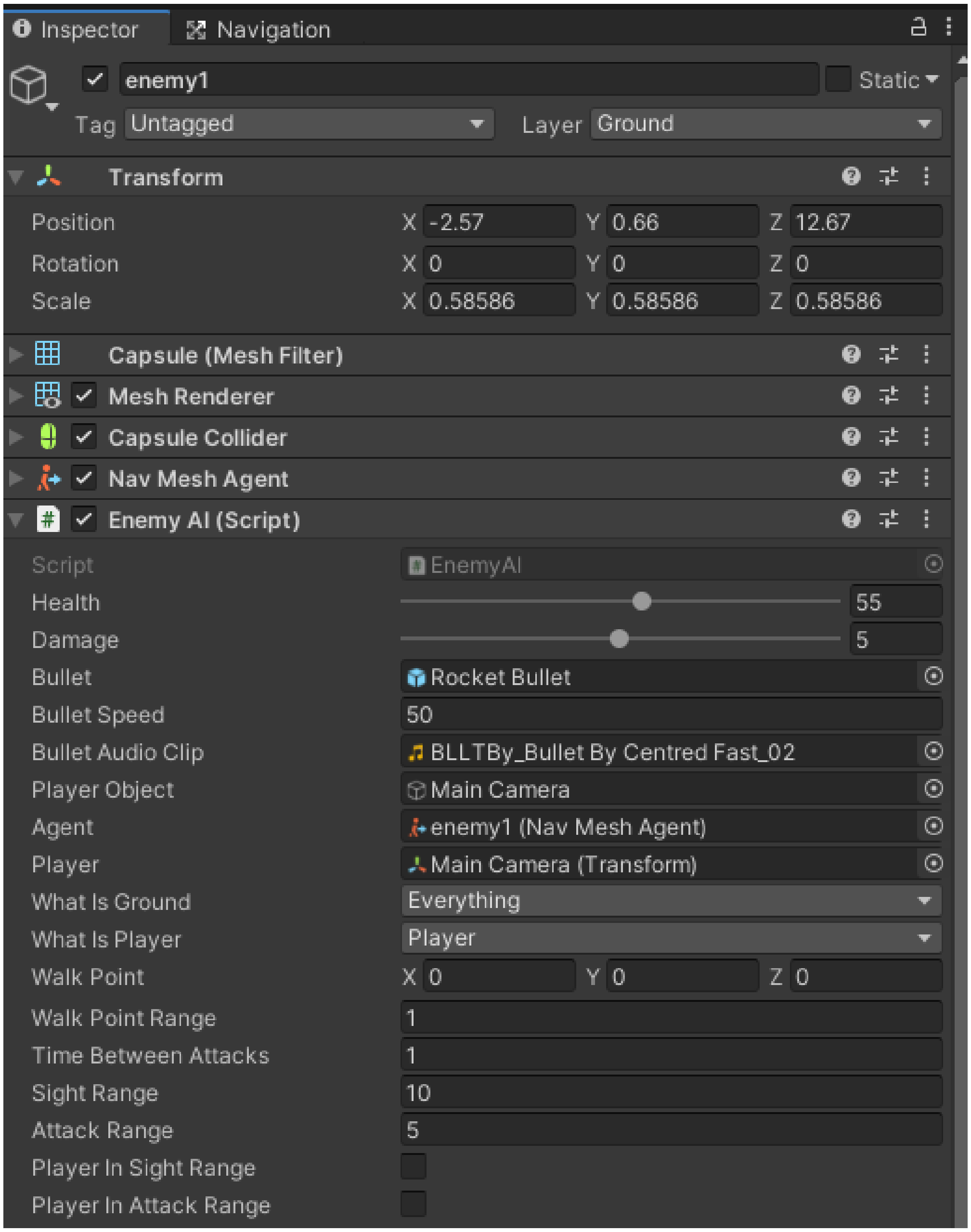
`agent` here is the NavMeshAgent passed in the script that calculates the next path to the player.

3. Attack
   - When chasing, if the player is in **Attack Range** the enemy is attacked with a bullet.

```
 private void AttackPlayer()
{
    agent.SetDestination(transform.position);
    transform.LookAt(player);
    if (!alreadyAttacked)
    {
        // Attack here
        GameObject spawnedBullet = Instantiate(bullet, agent.transform.position, agent.transform.rotation);
        // To avoid collision within itself put it in the player layer
        spawnedBullet.layer = 12;

        // Provide rigid body to the instantiated bullet
        if (spawnedBullet.GetComponent<Rigidbody>() != null)
        {
            spawnedBullet.GetComponent<Rigidbody>().velocity = bulletSpeed * transform.forward;
        }
        else
        {
            spawnedBullet.AddComponent<Rigidbody>().velocity = bulletSpeed * transform.forward;
        }

        spawnedBullet.AddComponent<BulletDamage>().bulletDamage = 10;
        spawnedBullet.GetComponent<BulletDamage>().target = playerObject;

        // Play attack sound
        AudioSource audioSource = spawnedBullet.AddComponent<AudioSource>() as AudioSource;
        audioSource.PlayOneShot(bulletAudioClip);
        Destroy(spawnedBullet, 2);
        alreadyAttacked = true;
        Invoke(nameof(ResetAttack), timeBetweenAttacks);
    }
}
```

A few things are happening, here but the most important is that a `spawnedBullet` is created when the state is in attack. This component has a rigid body to simulate physics effects and also includes `AudioSource` to generate the bullet sound in the game.
This attack is similar to the player weapon script.

Below is the script component attached to the enemy object.



☑ enemy1 □ Static

Tag Untagged Layer Ground

Transform

Position X -2.57 Y 0.66 Z 12.67

Rotation X 0 Y 0 Z 0

Scale X 0.58586 Y 0.58586 Z 0.58586

Capsule (Mesh Filter)

☑ Mesh Renderer

☑ Capsule Collider

☑ Nav Mesh Agent

☑ Enemy AI (Script)

Script EnemyAI

Health 55

Damage 5

Bullet Rocket Bullet

Bullet Speed 50

Bullet Audio Clip BLLTBy_Bullet By Centred Fast_02

Player Object Main Camera

Agent enemy1 (Nav Mesh Agent)

Player Main Camera (Transform)

What Is Ground Everything

What Is Player Player

Walk Point X 0 Y 0 Z 0

Walk Point Range 1

Time Between Attacks 1

Sight Range 10

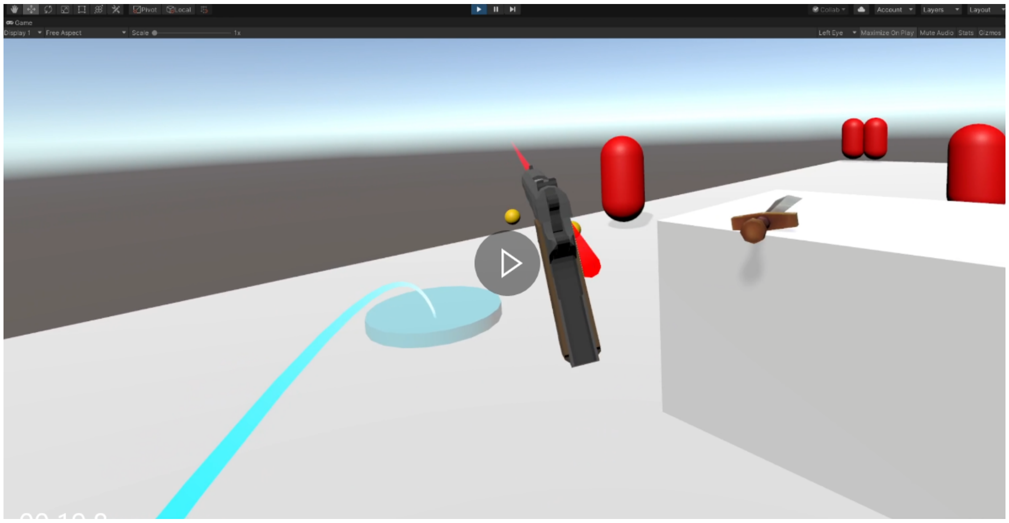
Attack Range 5

Player In Sight Range □

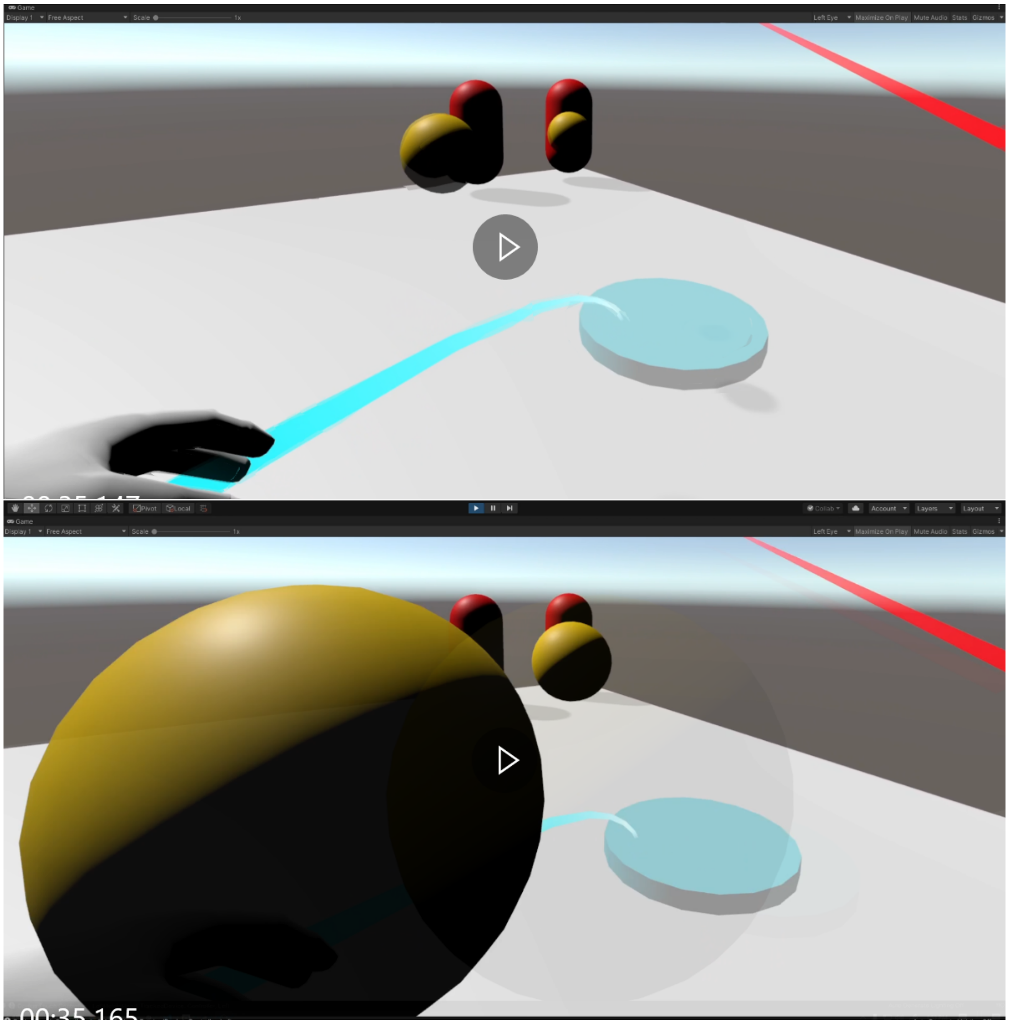
Player In Attack Range □

More on Enemy AI

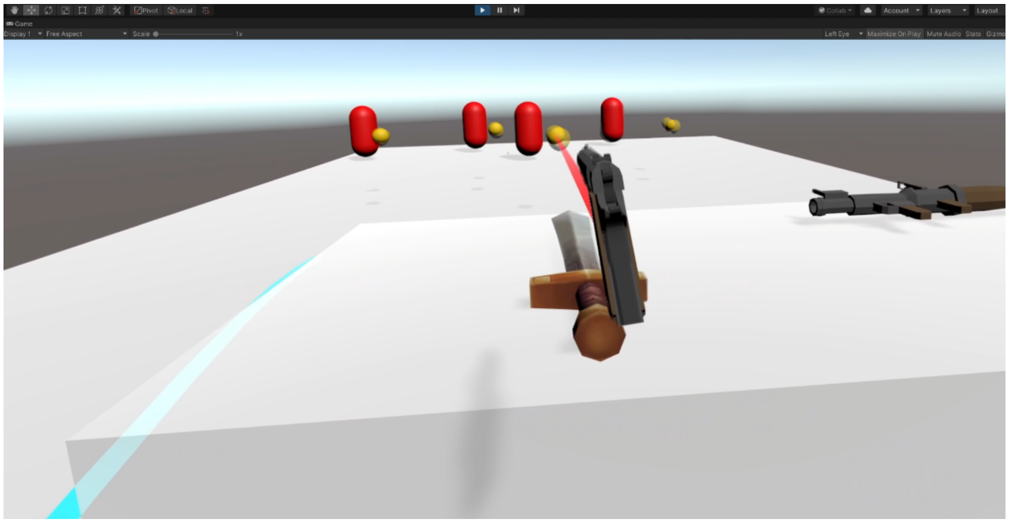
At this point my enemy AI are able to target the player and hit them as well. The chase works as intended. Enemy mesh is still work in progress but this will be implemented on the final map.



*Above is a demo of **chase** state we can see the enemies are getting closer to the player and yellow objects are shown to indicate the bullet.*



Above we can see projectiles from two enemies closely.



Above we can see projectiles being shot at the player. Since all the enemies are at range, all the enemies attack at a similar time.

Remaining challenges

The remaining challenge is just to implement enemy mesh and a dungeon map for the player to walk in. Additionally, player and target health are a missing component to the game along with the bullet ammo.

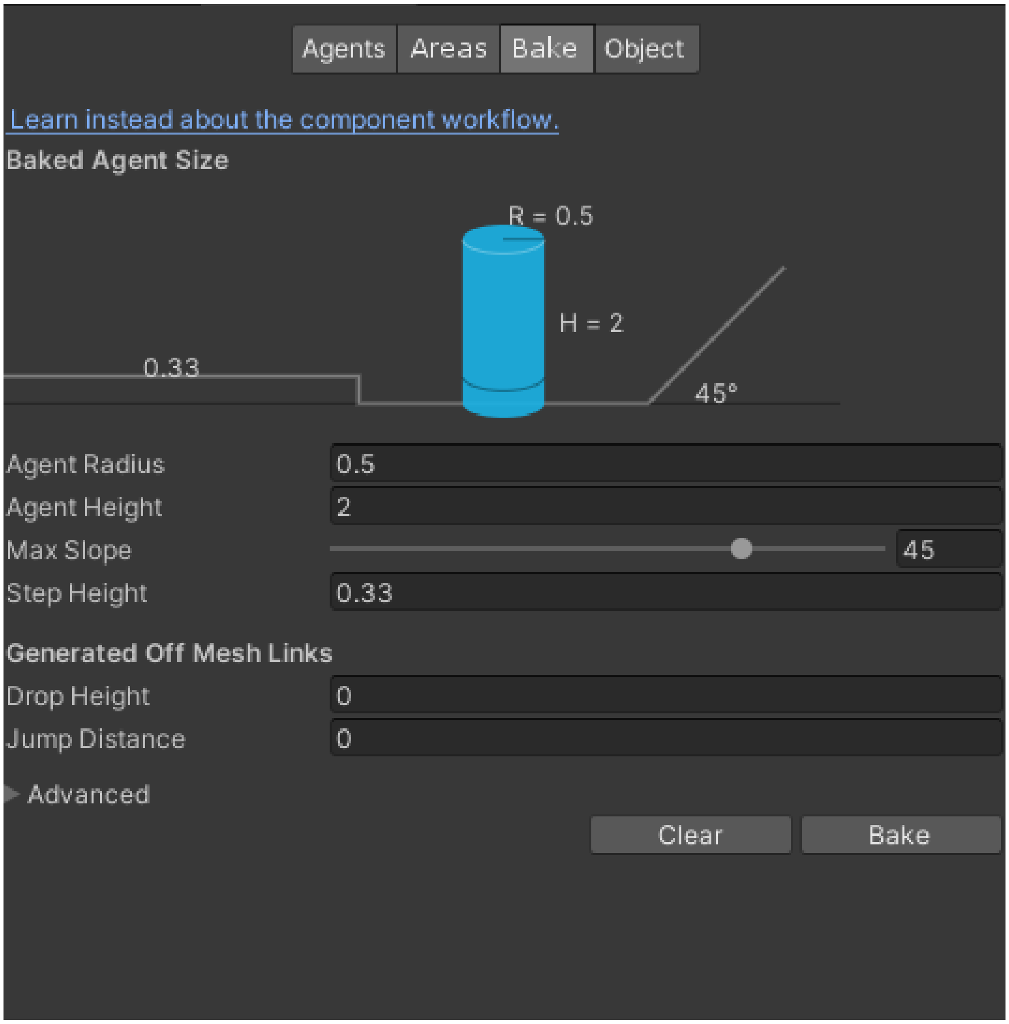
I would likely be working on projectiles more than phsyical attacks to make it easier on the player. So the game, will be a dungeon with guns instead of swords.

Aside from this, the physics mechanics of the game is pretty much working and designing the mesh, and map will be the remaining challenge.

Learning/Collaboration

- What are the new skills you gained so far as you built your VR app? Provide details including the development tools, concepts, and procedures you could easily explain or teach to the beginners.

The new skills I have gained was undoubtedly learning C# in Unity. One of the most significant aspect was dealing with AI and handling collisions that is triggered by the user. The tools I specifically used to handle AI is NavMeshAgent. This class in the UnityEngine.AI helps simplify the process of generating paths for the AI to follow, however, executing the paths still requires writing a script.
For example, below is baking a naigation for the floor. We can specify the properties of the "NavMeshAgent" which in this case is the red enemies.



Learn instead about the component workflow.

Baked Agent Size

Agent Radius 0.5

Agent Height 2

Max Slope 45

Step Height 0.33

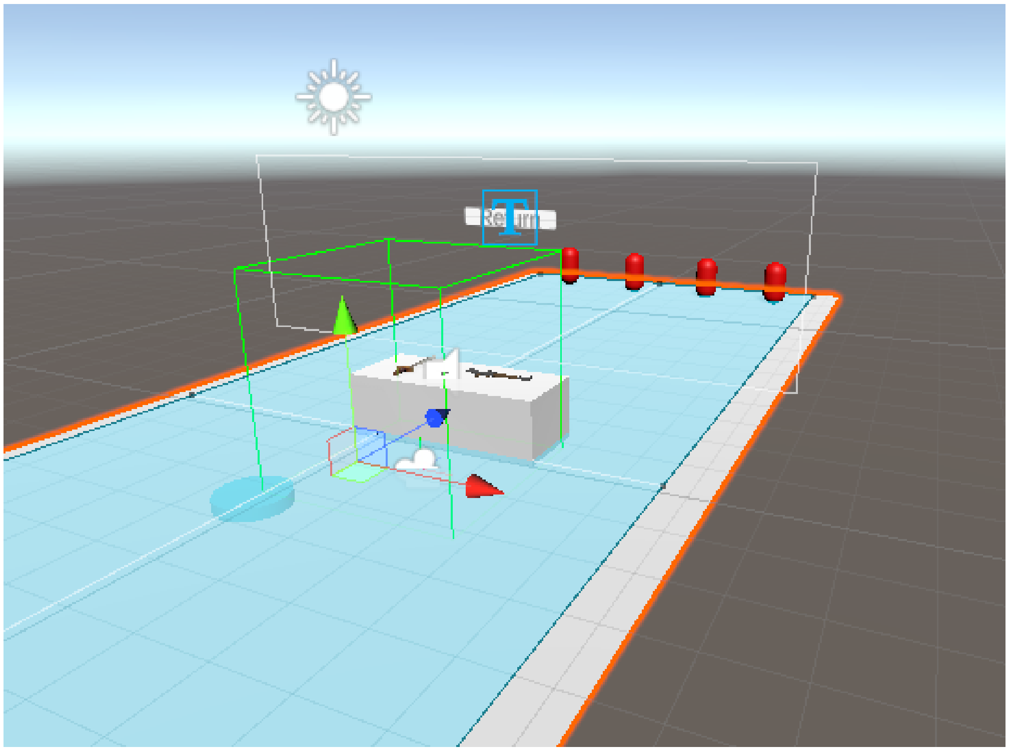
Generated Off Mesh Links

Drop Height 0

Jump Distance 0

Advanced

Proceeding to bake we get the following blue flooring below. This flooring indicates the Navigation area for the NavMeshAgents to move around or calculate the next step.



- How are you helping your team in the areas you became knowledgeable?
  For Bryan in my team I have suggested perhaps implementing a joystick for movement instead of constant movemnt of the head would be a good solution to prevent motion sickness from the movement. For Giana on the other hand, during group discussions I have provided some ideas on setting up her VR through Youtube tutorials such as Valem and choices for a VR device.
- Who helped you in your team or other teams if you received help or a great tip?
  A great tip was not to include reload. Since we're focusing on user experience and comfortability, I think mitigating the extra steps to enjoy a VR game is important for a fun game. Therefore, the weapons are infinite.
- What was your collaboration like with your teammates and class?
  Collaboration with my teammates proved to be helpful, and I enjoy the process of sharing and exploring potential weaknesses to my design and their designs. This in turn help me gain some user perspective that benefited my development of my application and what not implement that will hinder a great user experience.