VR Zombie Apocalypse Simulator

Jeet Avasare (21124021)

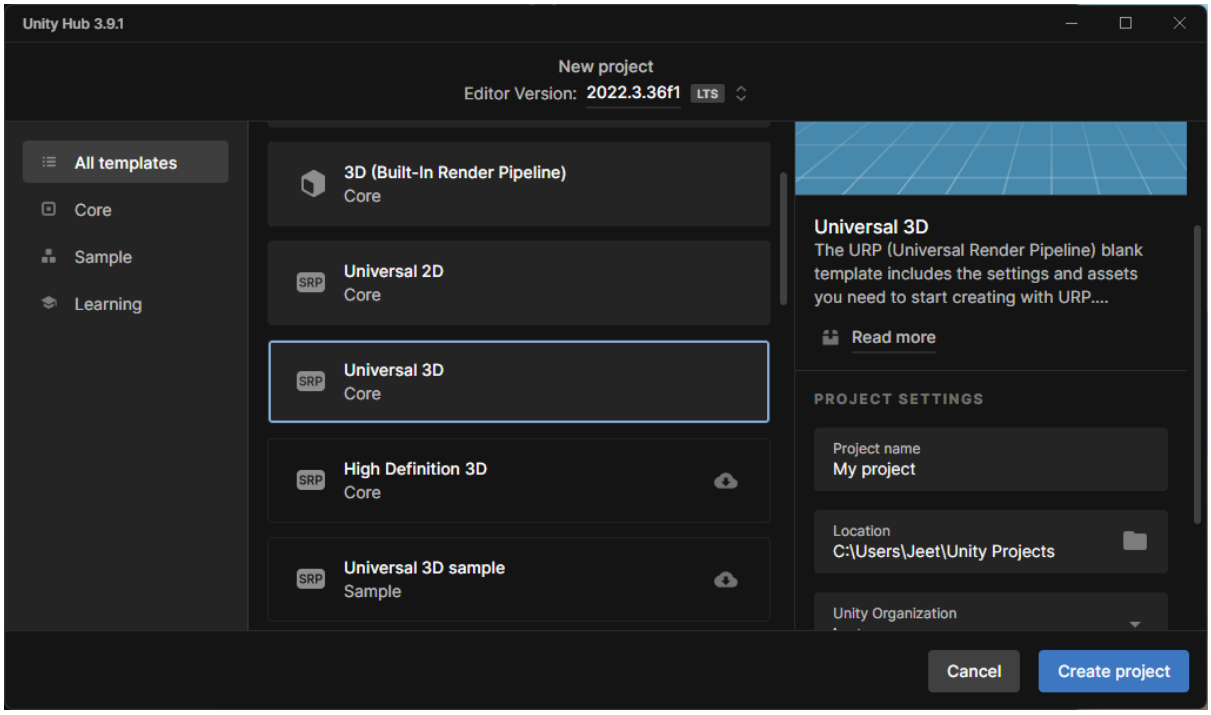
Navrachana University

26-10-2024

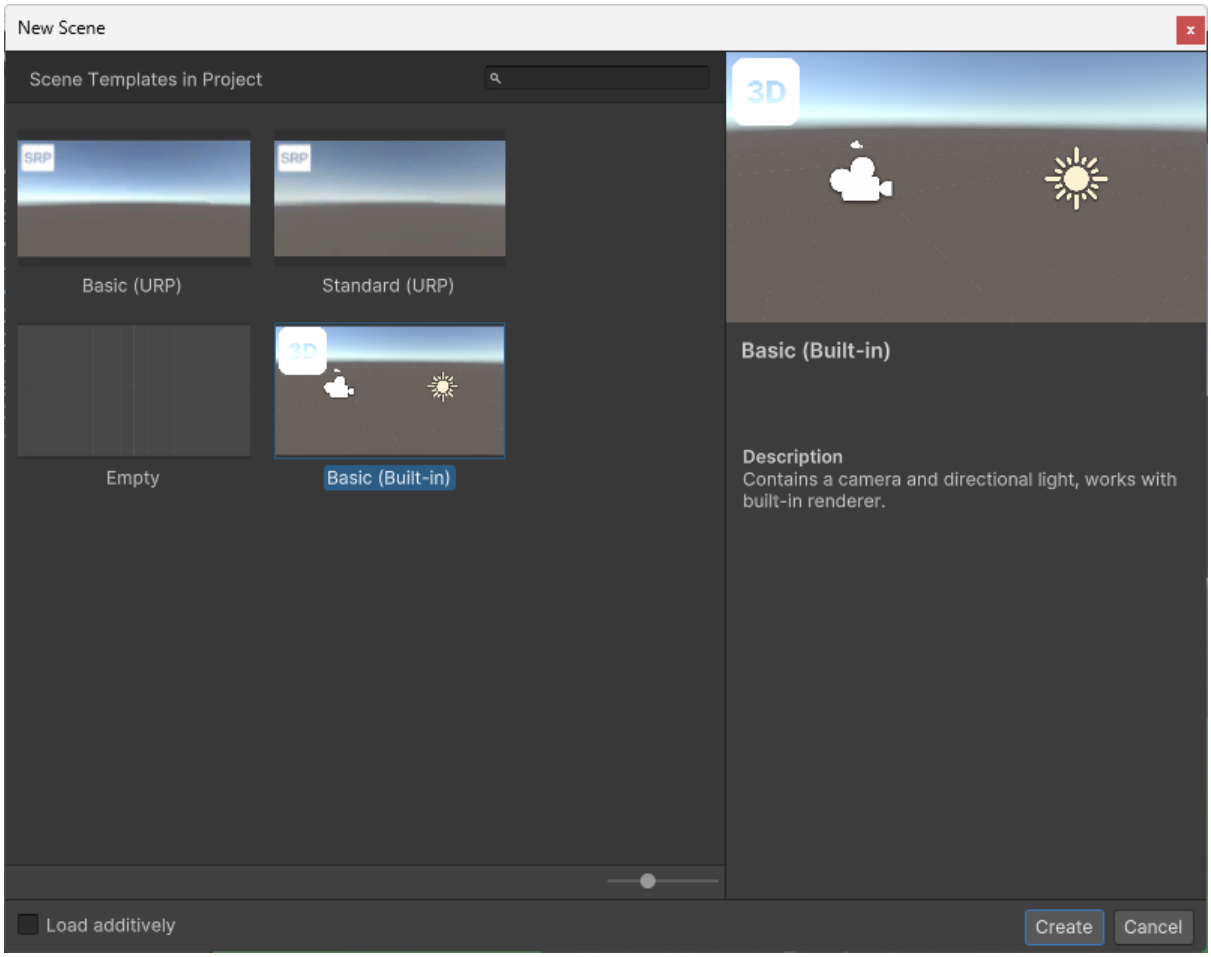
Task 1: Set Up Your Unity Project & Configure the VR Environment

Part 1: Creating the Project

Step 1: Create a new Unity project with 3D URP template



Step 2: Create a new Scene where we will be working on the VR setup

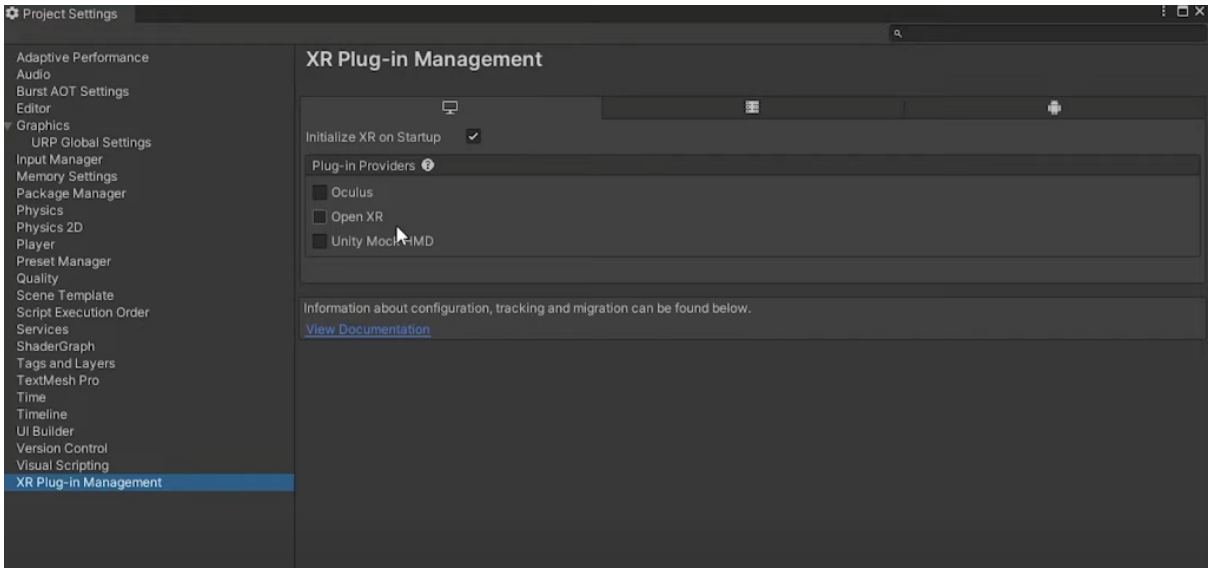

New Scene
Scene Templates in Project
SRP
Basic (URP)
SRP
Standard (URP)
Empty
3D
Basic (Built-in)
3D
Basic (Built-in)
Description
Contains a camera and directional light, works with built-in renderer.
Load additively
Create
Cancel

Part 2: Configuring VR Environment

Step 1: Install XR Plugin Management

Goto Project Settings -> XR Plugin Management

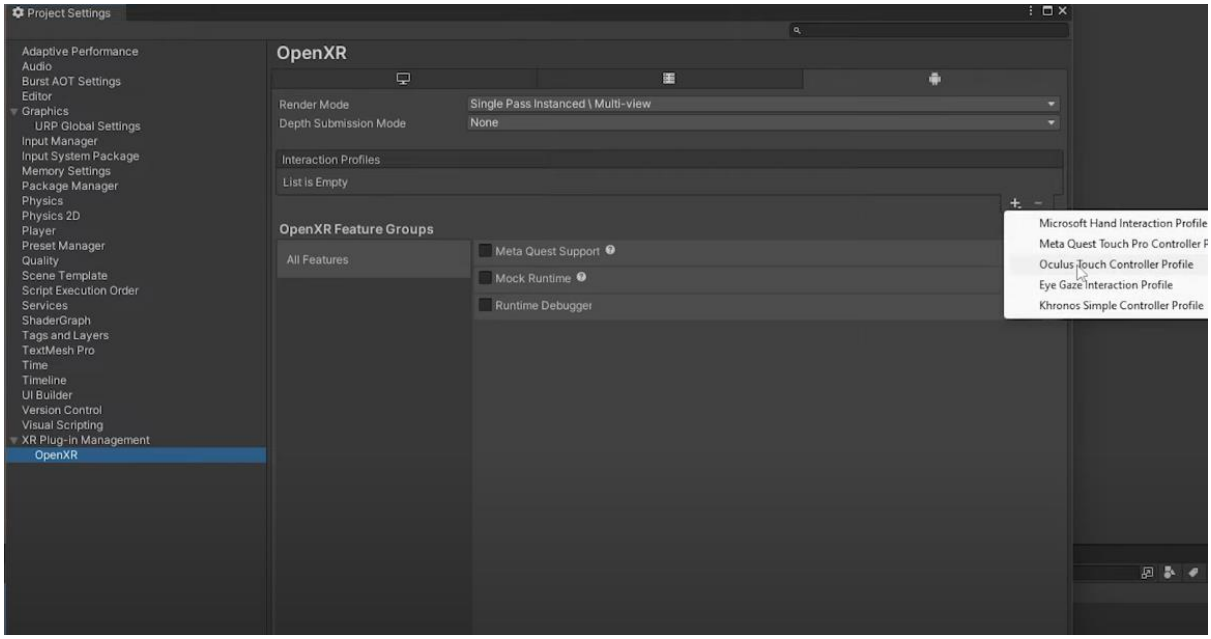
Step 2: After that's done on PC tab check Open XR check box under the plug in providers



Step 3: Expand XR Plugin Management and click Open XR

Under that find Interaction profiles and add profiles where you want to run and configure your VR Project

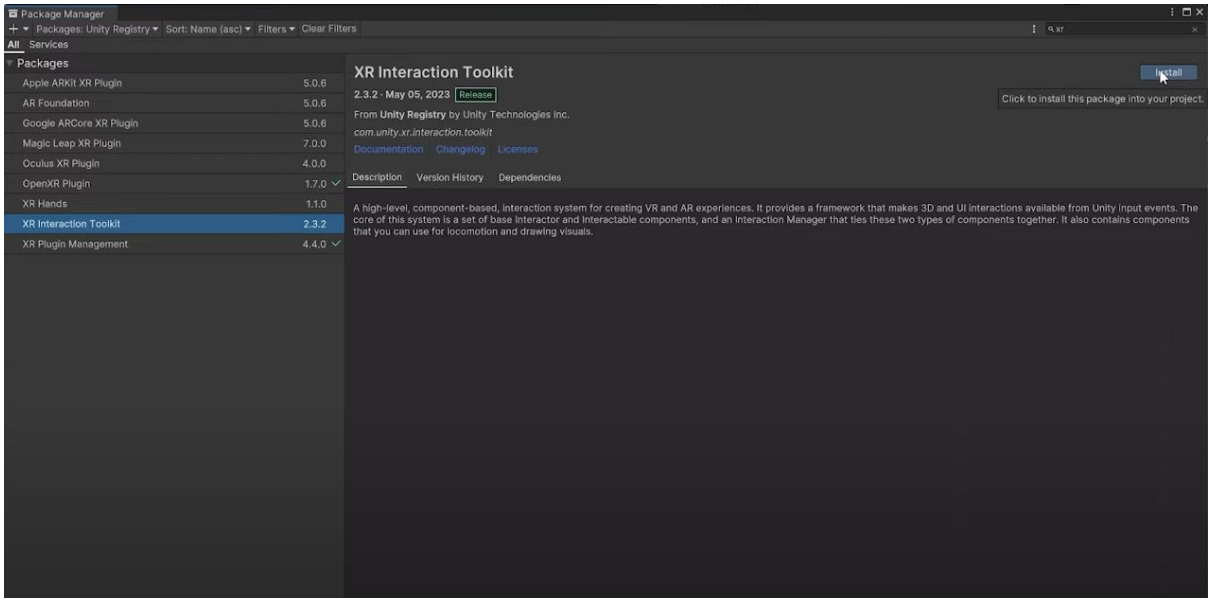
Some common profiles you might want to add are Oculus Touch controller profile, Meta Quest Touch



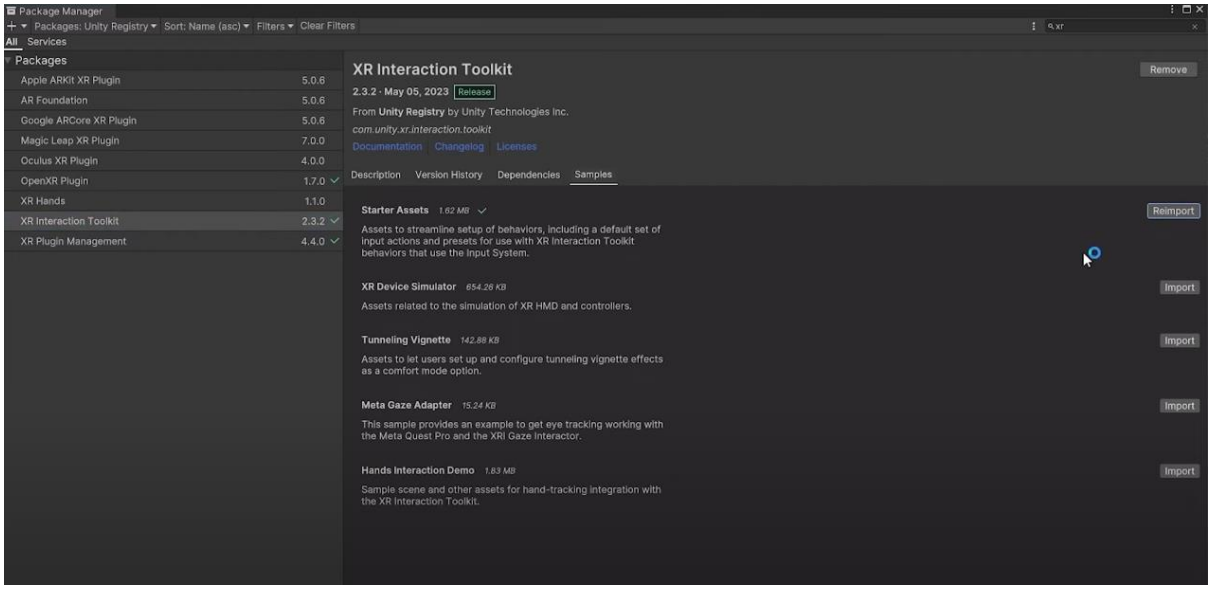
After that's done close the project settings window

Step 4: Open Window-> Package Manager

Inside Unity Registry Search for XR Interaction Toolkit and install it

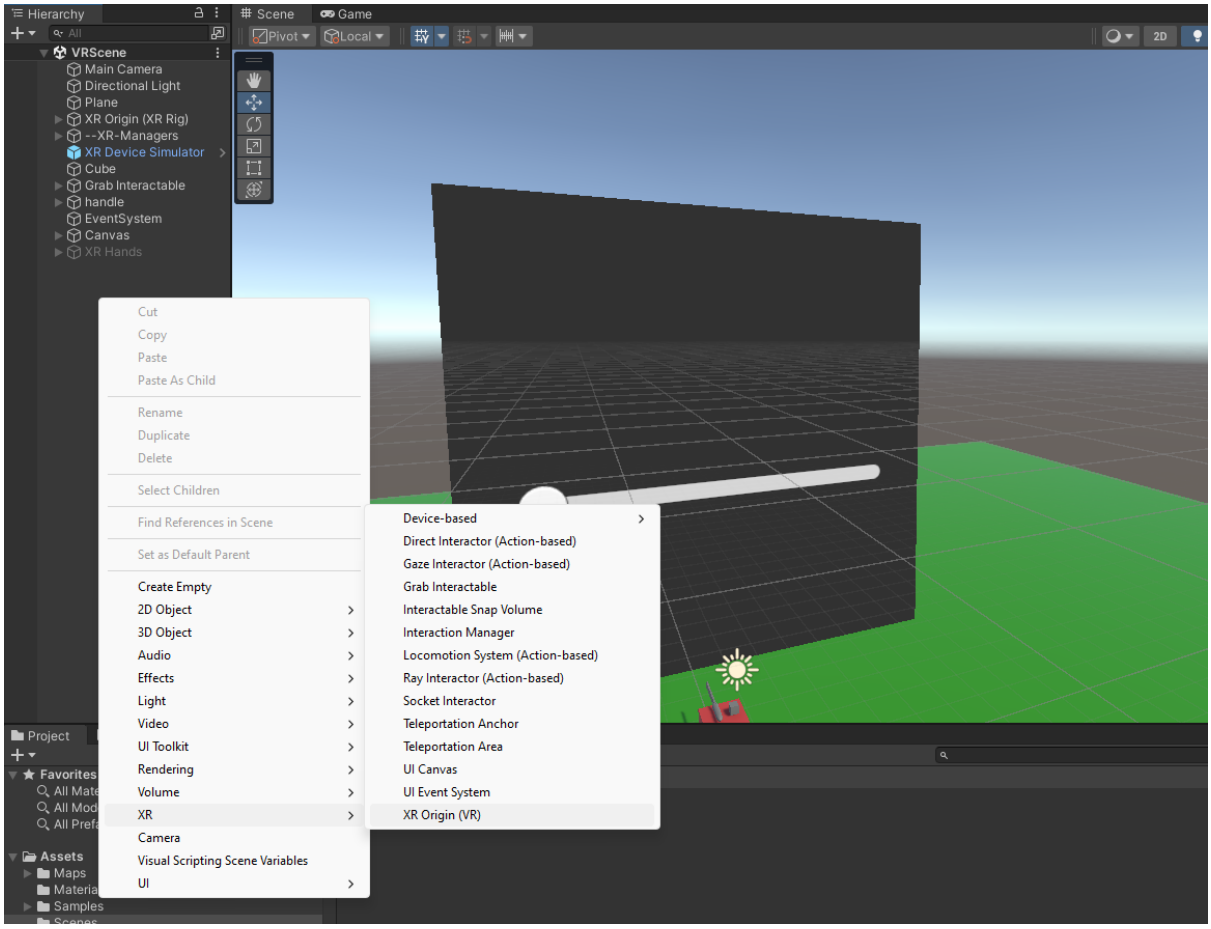


Step 5: After Installation Goto Samples tab and import Starter Assets, XR Device Simulators, Hands Interaction Demo



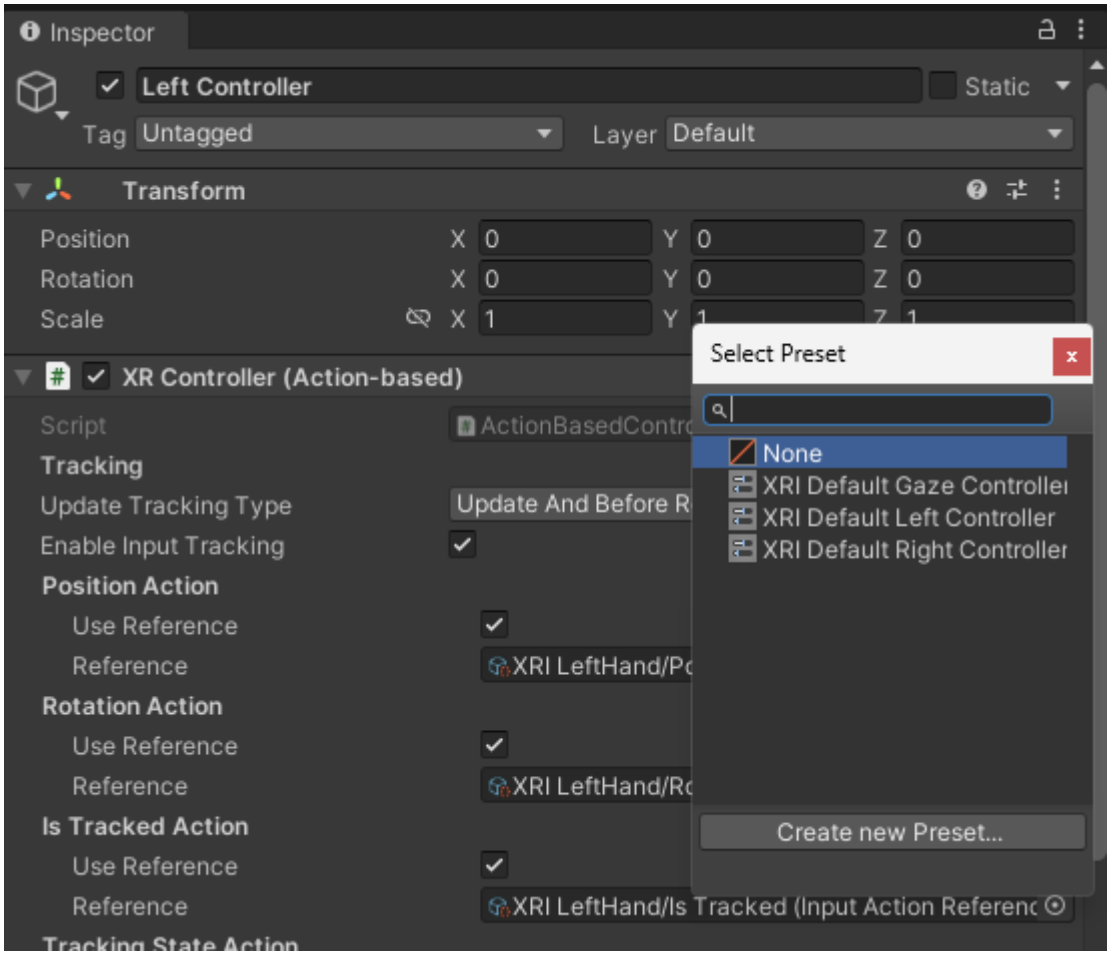
Step 6: Add XR Origin

Right Click on Hierarchy-> XR -> XR Origin (VR)



Step 7: Expand the XR Origin (XR Rig)

Find Left Controller and on the inspector tab find XR Controller (Action based) and click on the Slider Icon and select *XRI Default Left Controller* preset



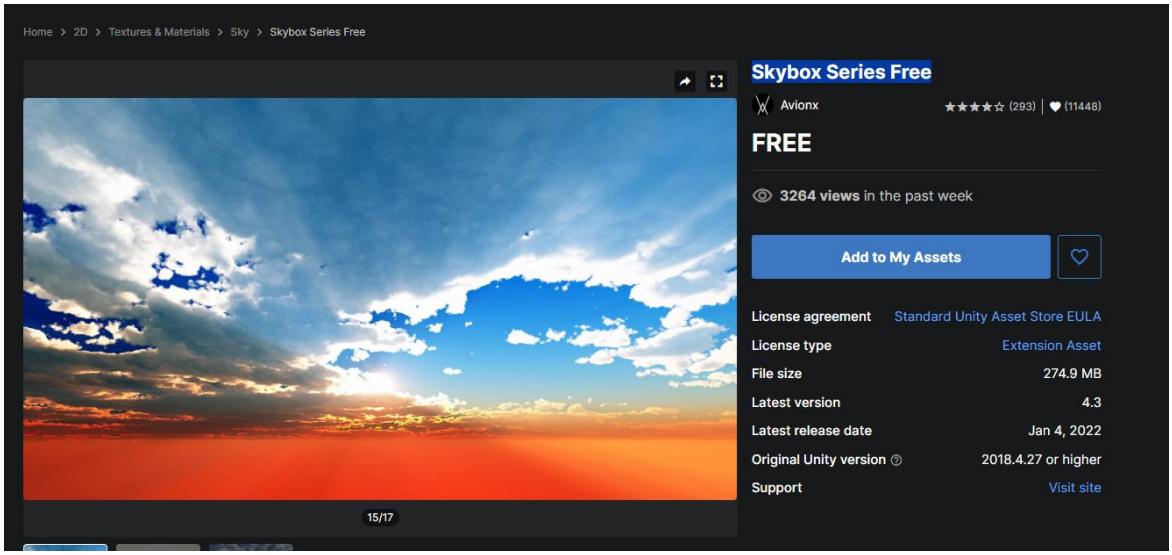
Similarly do for the right controller as well

Step 8: Add a simple plane for reference

Task 1 is now complete and you might boot up the scene and check out the movement controls, you can do it by connecting the VR headset or if you don't have one you can still move around without your keyboard thanks to XR device simulator

Task 2: Create the Ground Plane and add a Skybox

Part 1: Adding a sky box



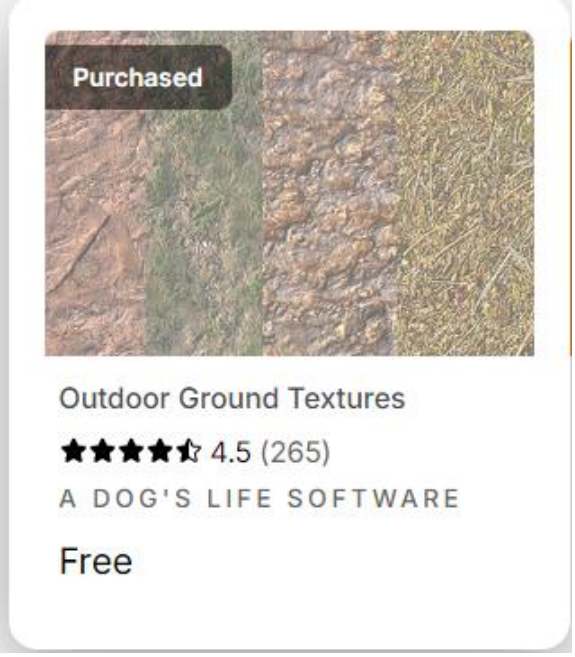
Import this asset

And on the sky drag and drop the skybox onto the sky

Part 2: Adding Ground

Step 1: Assets to be downloaded

All the below assets are free to download sources in references



Conifers [BOTD]

★★★★⯪ 4.4 (130)

FORST

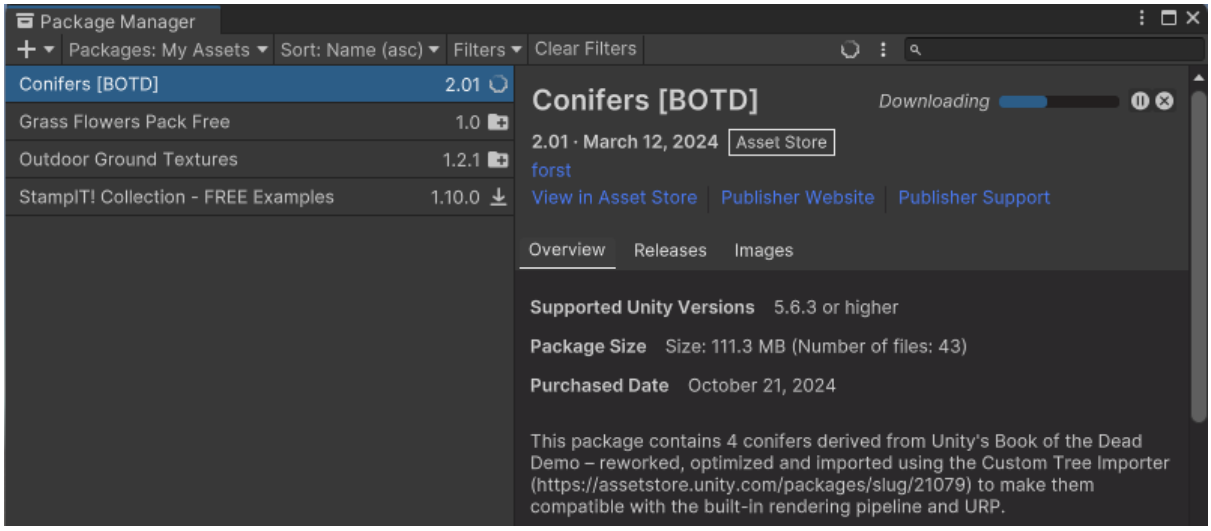
Free

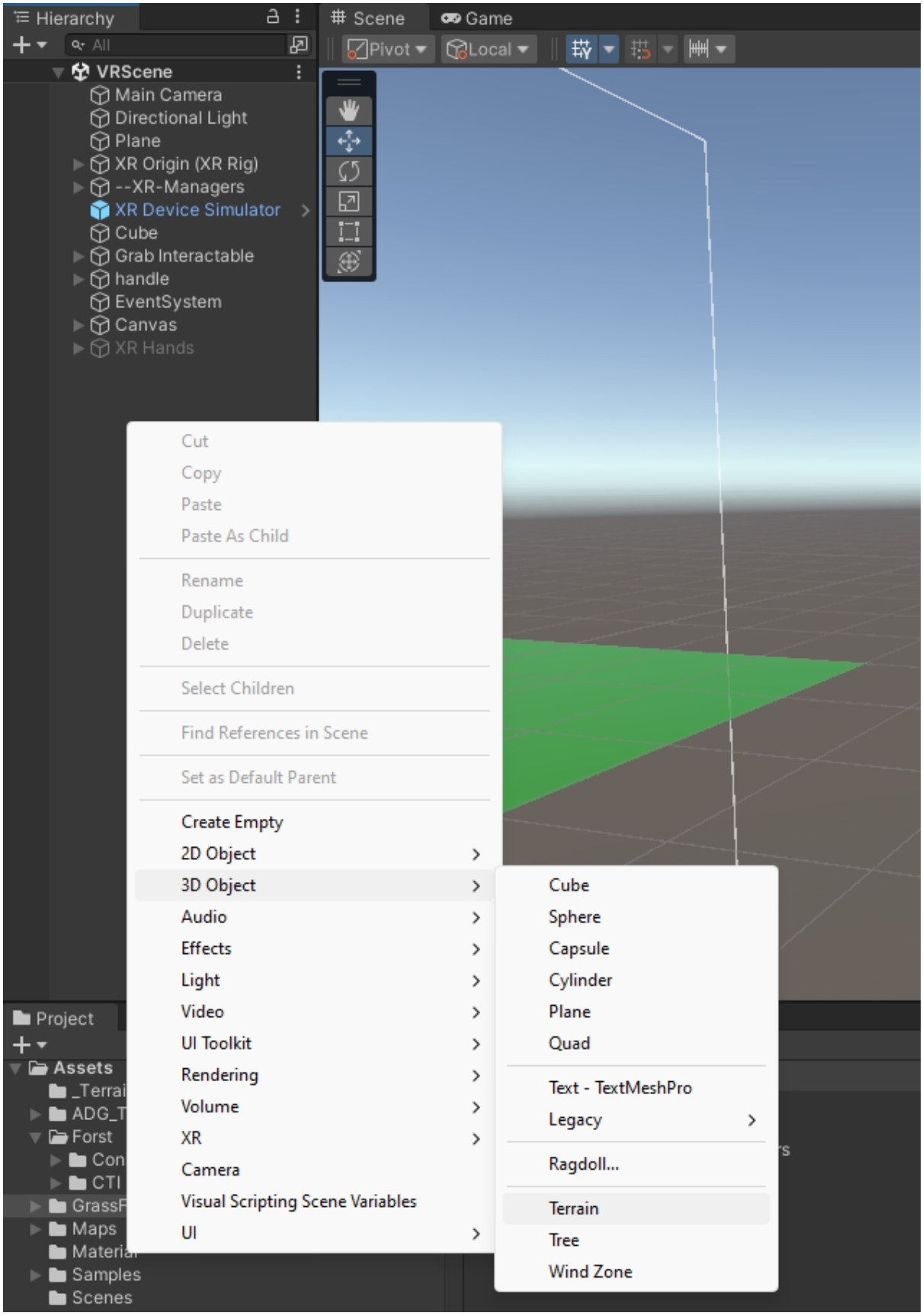
Grass Flowers Pack Free

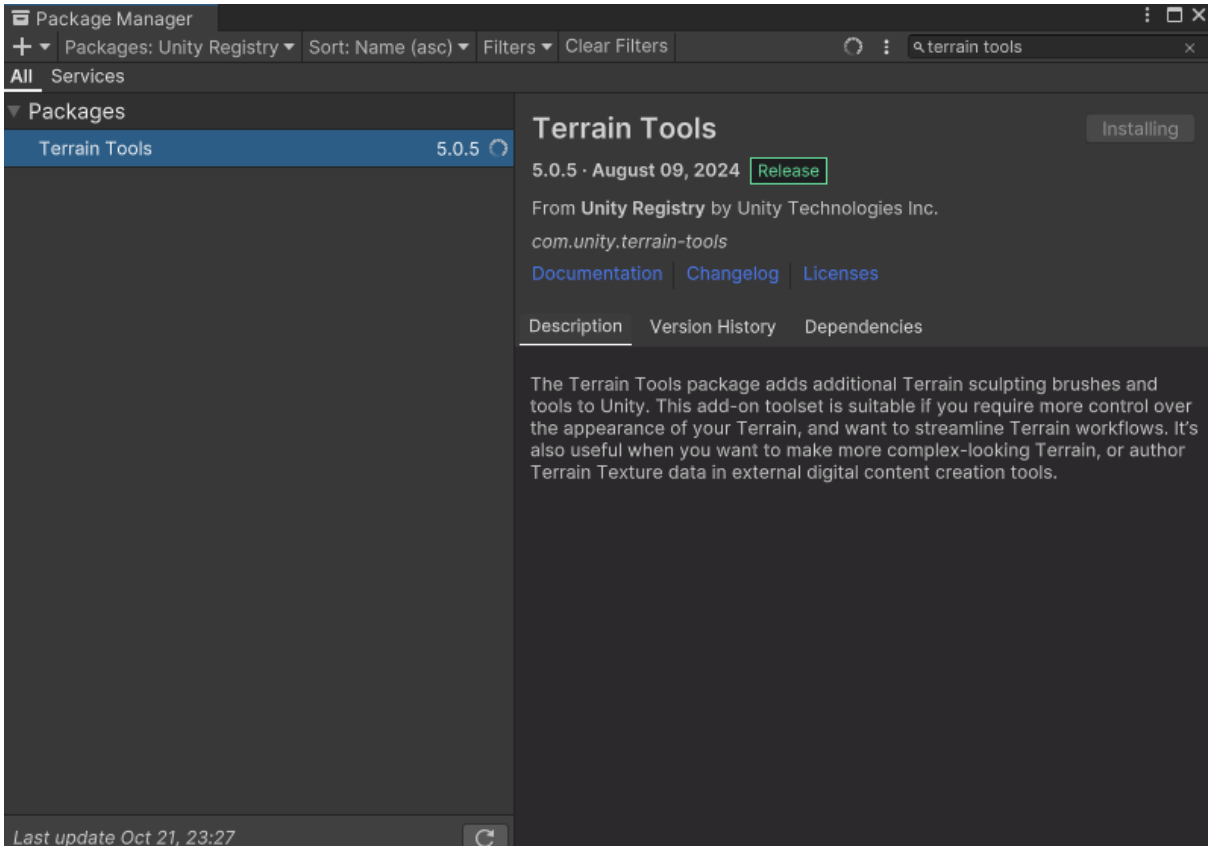
★★★★⯪ 4.7 (128)

ALP

Free


Package Manager
Packages: My Assets
Sort: Name (asc)
Filters
Clear Filters
Conifers [BOTD] 2.01
Grass Flowers Pack Free 1.0
Outdoor Ground Textures 1.2.1
StampIT! Collection - FREE Examples 1.10.0
Conifers [BOTD]
Downloading
2.01 · March 12, 2024
Asset Store
forst
View in Asset Store
Publisher Website
Publisher Support
Overview
Releases
Images
Supported Unity Versions 5.6.3 or higher
Package Size Size: 111.3 MB (Number of files: 43)
Purchased Date October 21, 2024
This package contains 4 conifers derived from Unity's Book of the Dead Demo – reworked, optimized and imported using the Custom Tree Importer (https://assetstore.unity.com/packages/slug/21079) to make them compatible with the built-in rendering pipeline and URP.


Hierarchy
Scene
Game
Pivot
Local
VRScene
Main Camera
Directional Light
Plane
XR Origin (XR Rig)
--XR-Managers
XR Device Simulator
Cube
Grab Interactable
handle
EventSystem
Canvas
XR Hands
Cut
Copy
Paste
Paste As Child
Rename
Duplicate
Delete
Select Children
Find References in Scene
Set as Default Parent
Create Empty
2D Object
3D Object
Audio
Effects
Light
Video
UI Toolkit
Rendering
Volume
XR
Camera
Visual Scripting Scene Variables
UI
Cube
Sphere
Capsule
Cylinder
Plane
Quad
Text - TextMeshPro
Legacy
Ragdoll...
Terrain
Tree
Wind Zone
Project
Assets
Samples
Scenes

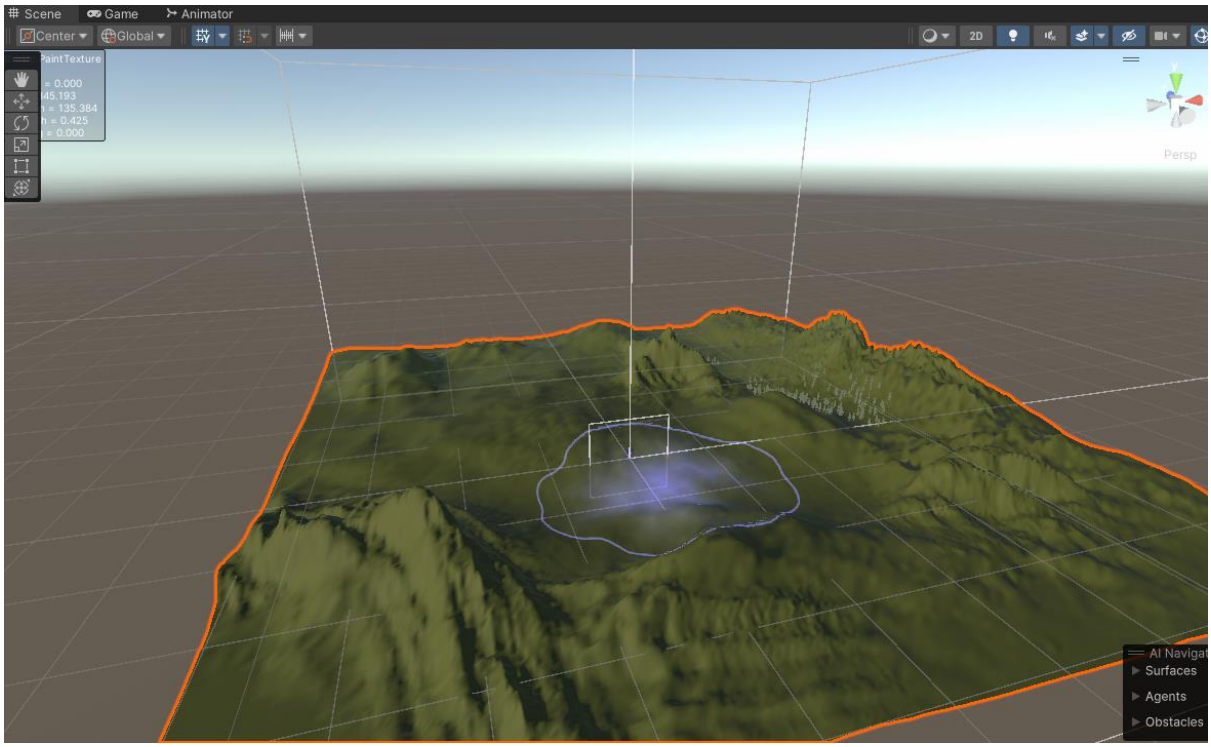


Step 2: Install and import the assets

Step 4: Using Terrain Brushes

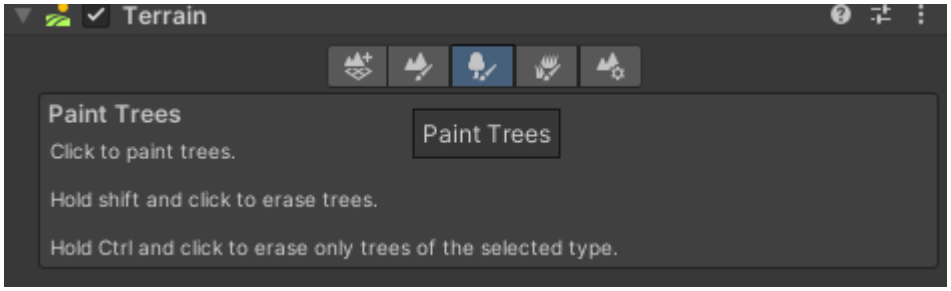
Select Raise or Lower height in terrain tools and disturb the terrain according to your liking

Below is the result



Task 3: Add Environment Objects

Step 1: Select the terrain object and select paint trees section

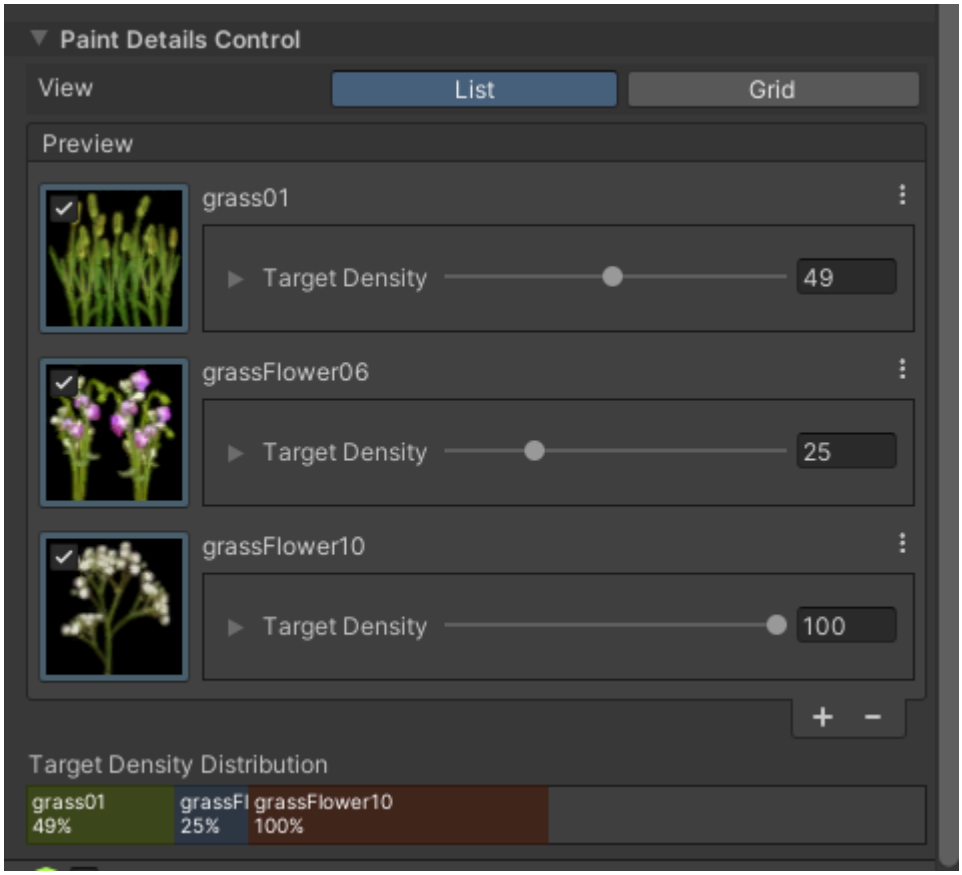


Step 2: Click on edit trees and search for the conifers trees asset we downloaded, add trees and adjust density according to your liking and pain the terrain with the trees

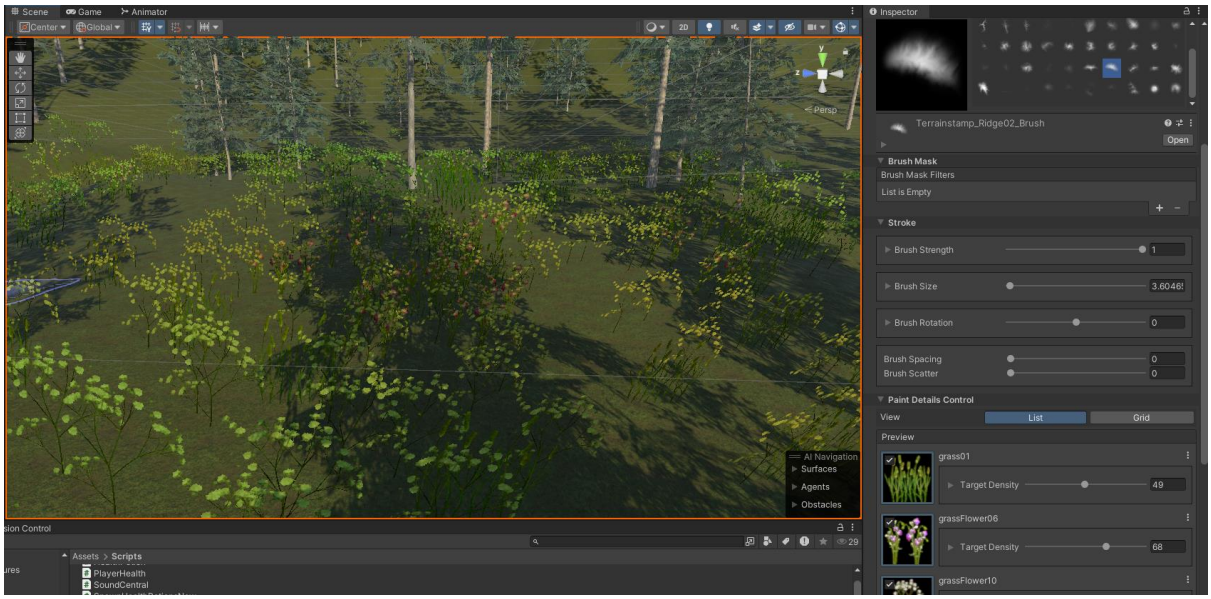
Step 4: Click on the paint details section and scroll down to Paint details Control

Click on the + icon and add grass and flower textures

You can change their relative densities according to your liking

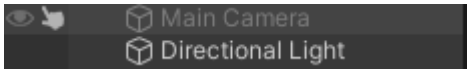


Step 5: Paint the terrain with details



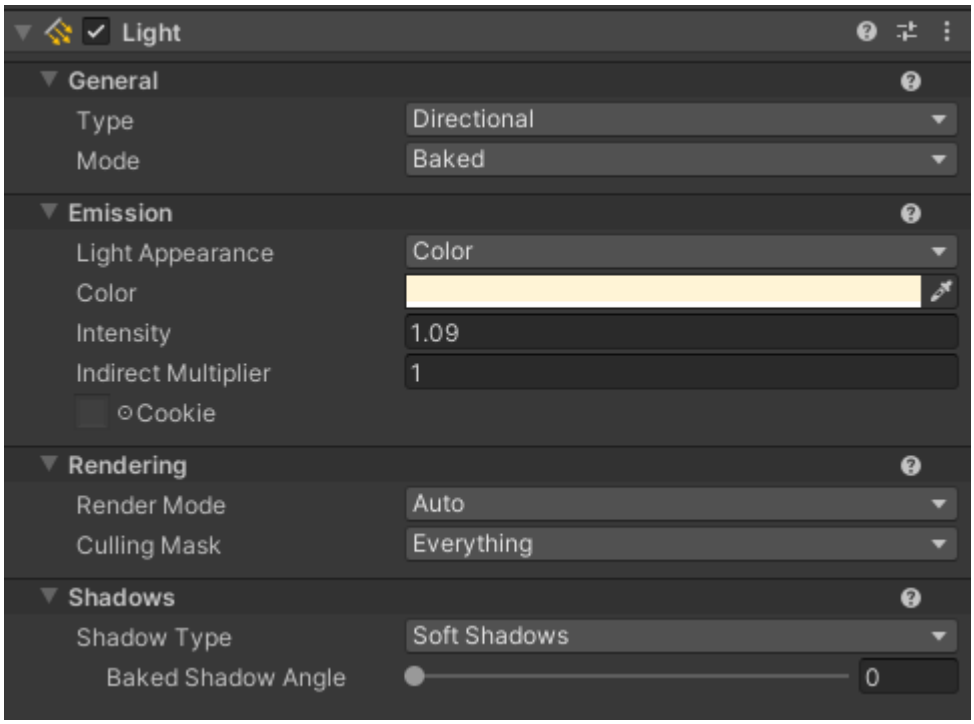
Task 4: Configure Lighting and Shadows

Step 1: Select the directional Light object in the hierarchy window



Step 2: Rotate the lighting according to the time of the day you want

Step 3: Adjust the colour temperature and intensity of the light according to your likng



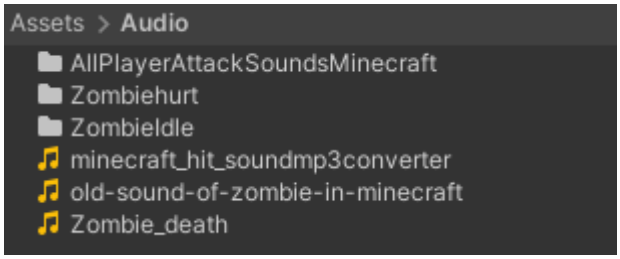
Step 4: Adjust how you want the shadows to be casted for the light

Select Soft Shadows for a balanced look

Task 5: Adding Audio

Step 1: Deciding and Collecting Sound Samples

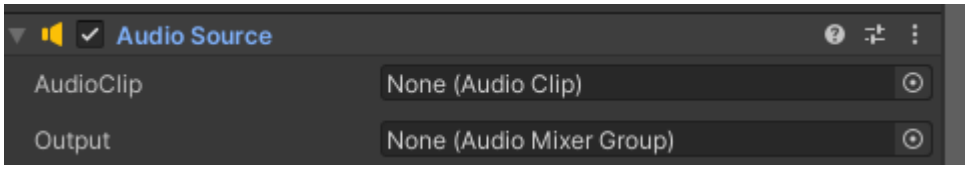
Zombie apocalypse project requires zombie sounds variations such as zombie idle sounds, zombie hurt sounds, zombie spawn sounds, zombie death sounds as well as player sounds like player hurt sound, player attacks sounds



For this project all of the sound's assets were extracted from the game of 'Minecraft'

Step 2: Playing the sounds

To play the sounds each object that will play the sounds will need to have the Audio Source Component, for now keep the Audio Clip section empty we will play sounds with the help of scripts



Step 3: Randomising Sounds

Now a single event can have multiple variations of sounds, we need to juggle between them randomly to give a alternation to sounds to give a more natural and immersive experience, for this we will call the below function

```
3 references
public static AudioClip PlayRandomSound(List<AudioClip> soundList)
{
    if (soundList.Count > 0)
    {
        int randomIndex = Random.Range(0, soundList.Count);

        return soundList[randomIndex];
    }
    else
    {
        Debug.LogWarning("Sound list is empty!");
        return null;
    }
}
```

Put this in a class `SoundCentral class` to better organise our code

The above script simply generates a random index between 0 and length of the List of Sound Variations and return a random sound based on index.

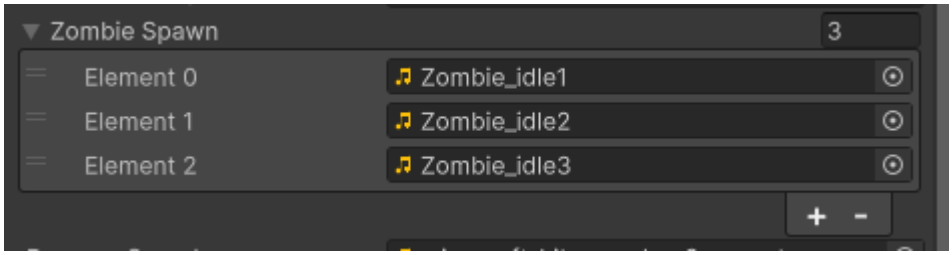
We can use the function to get a random sound from a list of sound variations on a particular event such as player attack event, player hurt event etc.

Step 4: We will assign variations by creating a List of AudioClips

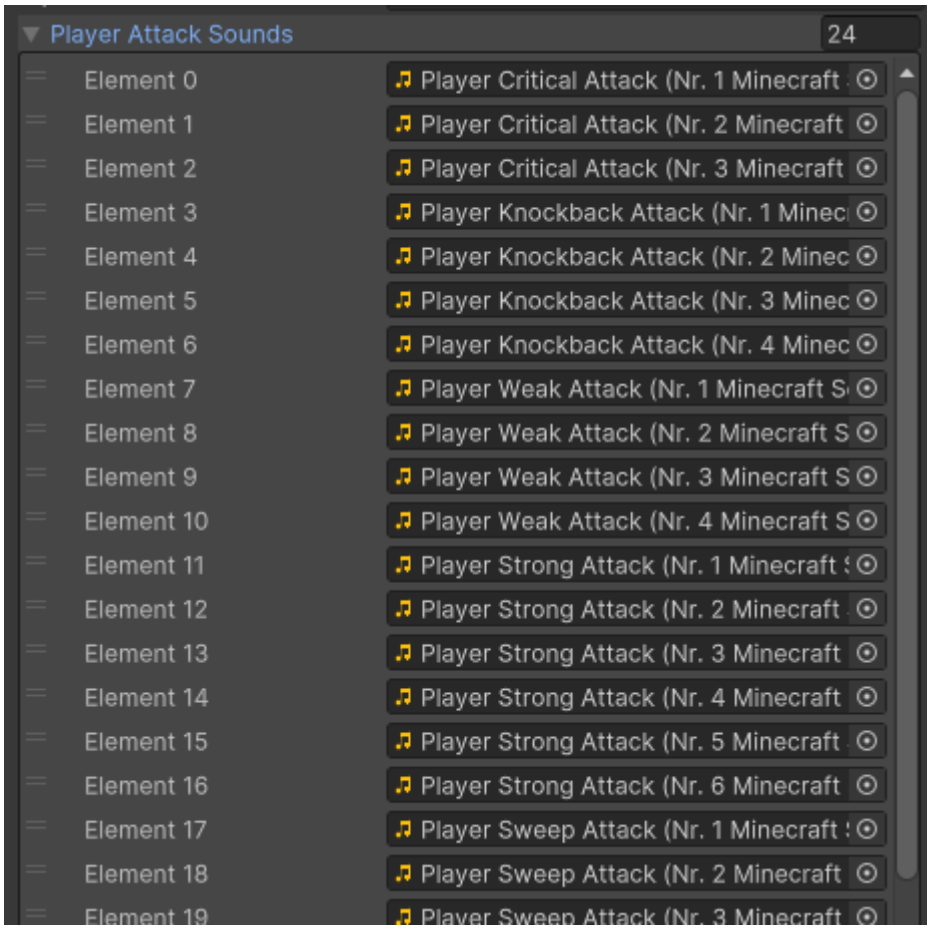
```
private AudioSource audioSource;
public List<AudioClip> zombieSpawn;
```

And assigning multiple audios for a single event in Unity Inspector window

E.g. 1



E.g. 2



Task 6: Implement Basic VR Interaction

Part 1: Cleaning up unnecessary defaults

Step 1: Go to left controller in our scene and except XR Controller (Action-based) and remove everything else like Line renderer, XR interactor line visualiser, sorting group

Similarly do for right controller as well

Part 2: Adding Direct Interactor to our hands

Now we want the grab the objects by reaching out to it and grabbing it with our palms

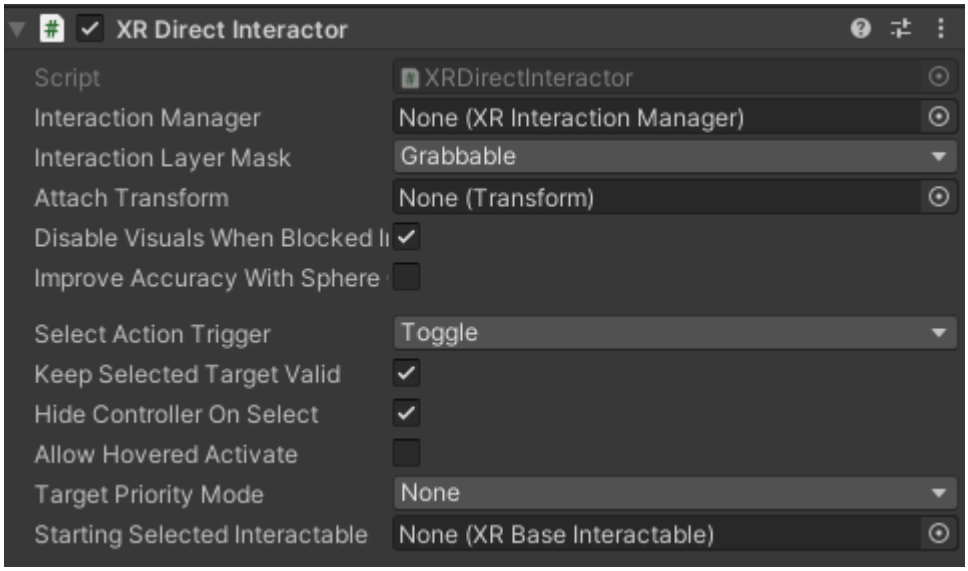
So, we only want to use Direct Interactor and not Ray interactor or Poke Interactor

Step 1: Go to left controller create a empty game object inside it and name it Direct Interactor

Step 2: Select the Direct Interactor and add XR Direct Interactor Script to it

Step 3: In its interaction layer mask create a new layer name it grabbable and choose it

Now only objects which have XR grab interactable script to it along with layer as grabbable will be the only objects the player can grab with left hand



Follow the similar steps in right hand controller or skip if you only want to grab the sword with left hand only

Part 3: Creating a locomotion system to allow the player to move around in the scene with controller

Step 1: In the XR Origin (XR Rig) game object create a empty game object name it Locomotion System.

Add to it the script **Locomotion System**

Step 2: Inside it add two more empty game objects and name it Move and Turn

Step 3: Adding Continuous Move provider

Select the Move object and add Continuous move provider (Action Based)

Fill and adjust the below fields in the scripts with the following content

System fields - Locomotion System game object we created above

Enable Strafe – Check

Use Gravity – Check

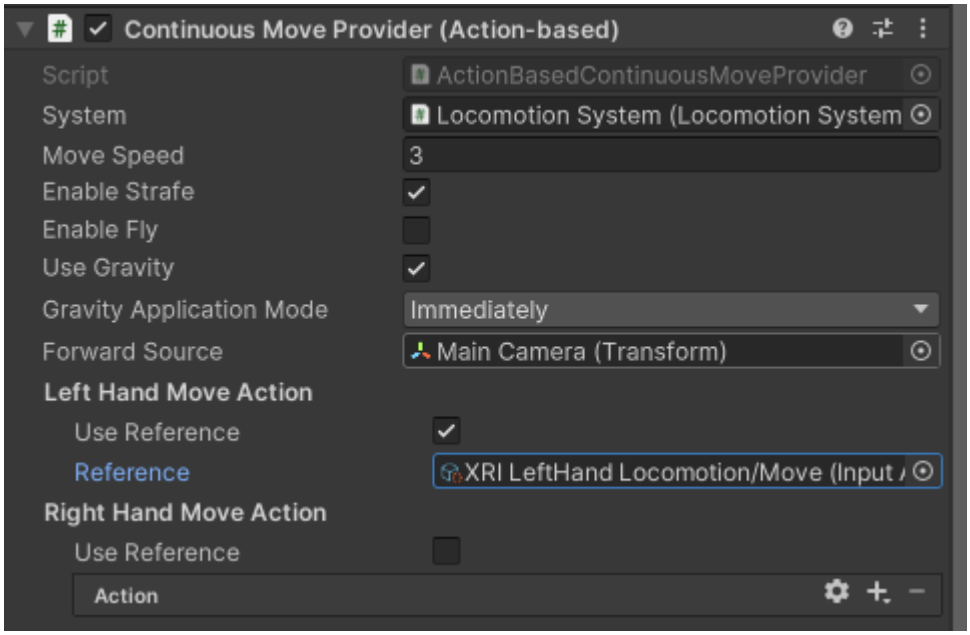
Gravity Application Mode - Immediately

Forward Source – Main Camera inside XR Origin -> Camera Offset -> Main Camera

Left Hand Move action – Check Use reference and set the reference as XRI LeftHand Locomotion/Move

Optionally you can increase the move speed to 3

After doing the component should look something like this



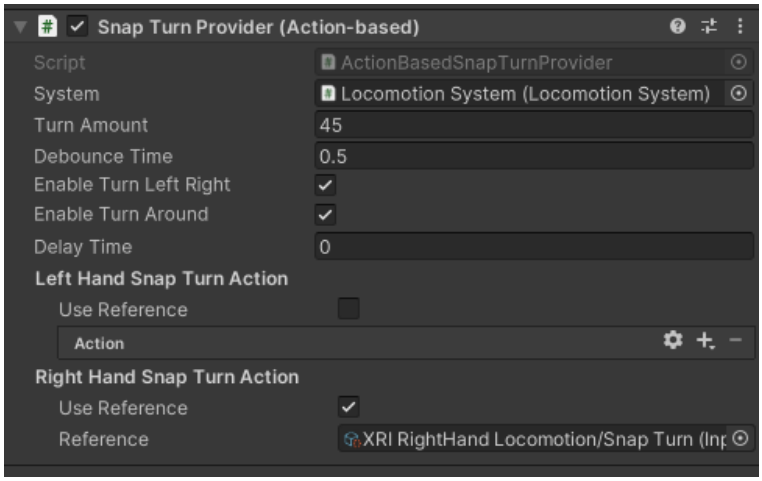
Step 4: Adding snap turn provider

Select the Turn object and add Snap Turn provider (Action Based)

Fill and adjust the below fields in the scripts with the following content

System fields - Locomotion System game object we created above

Left Hand Snap Turn action – Check Use reference and set the reference as XRI RightHand Locomotion/Turn



After following steps 3 and 4 player can move with his left hand controls and snap turn with his right hand controls

We added snap turn because some player might get dizzy from continuous turn which feels like a we are spinning in the scene which we don't want

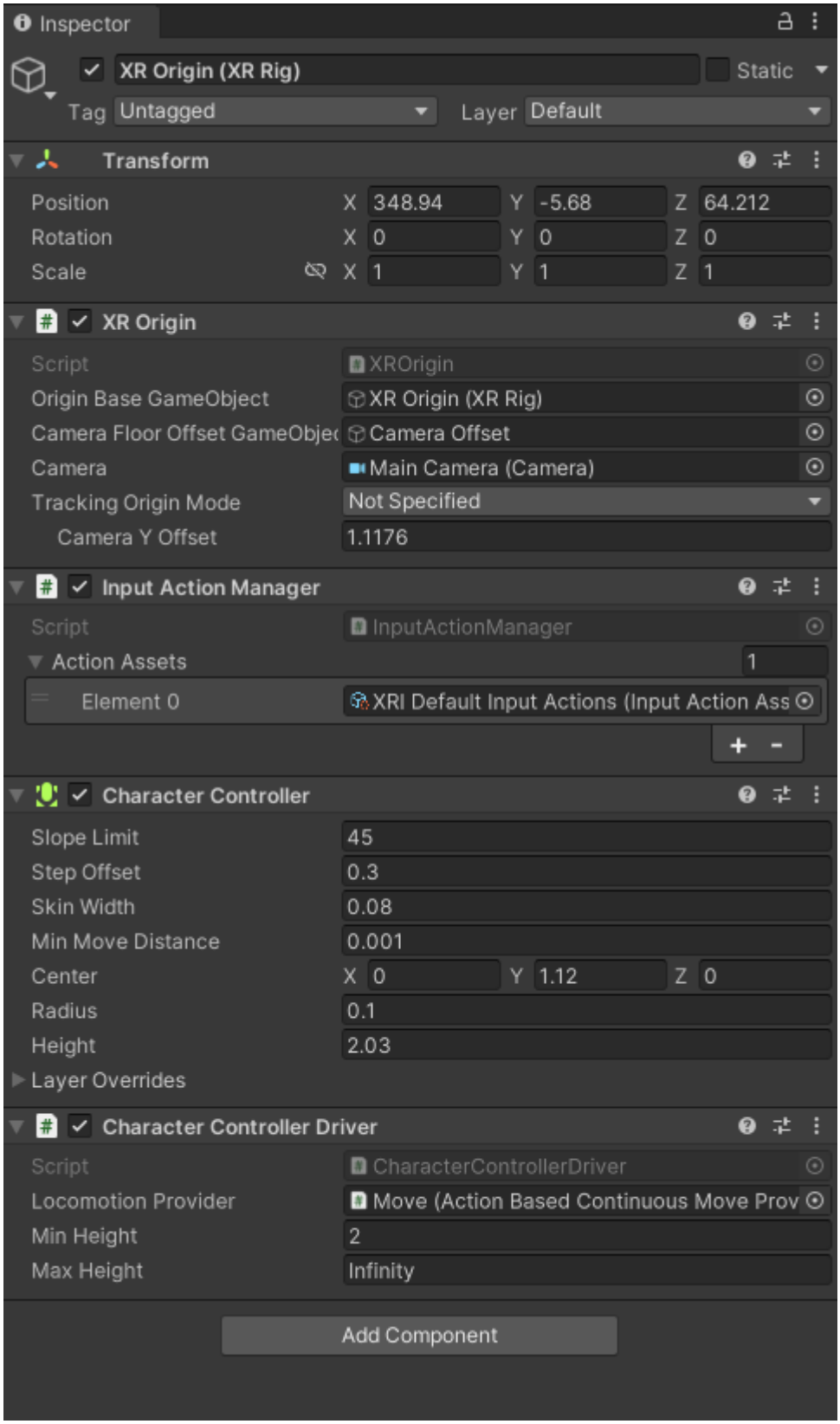
Part 4: Adding XR Interaction Manager

Create a empty game object name it –XR-Managers

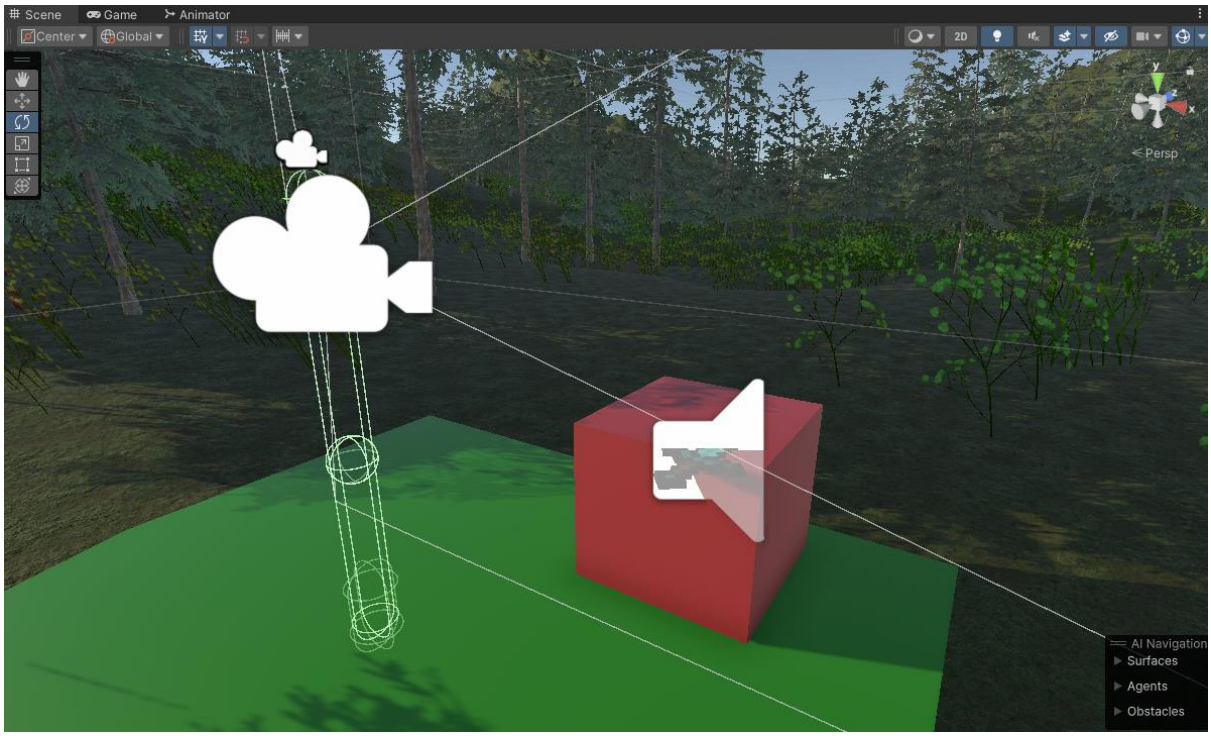
And inside it add the script XR Interaction Manager

Part 5: Making the XR rig a character

To the XR Origin game object add Character Controller and Character Controller Driver and fill out the fields like in the below image

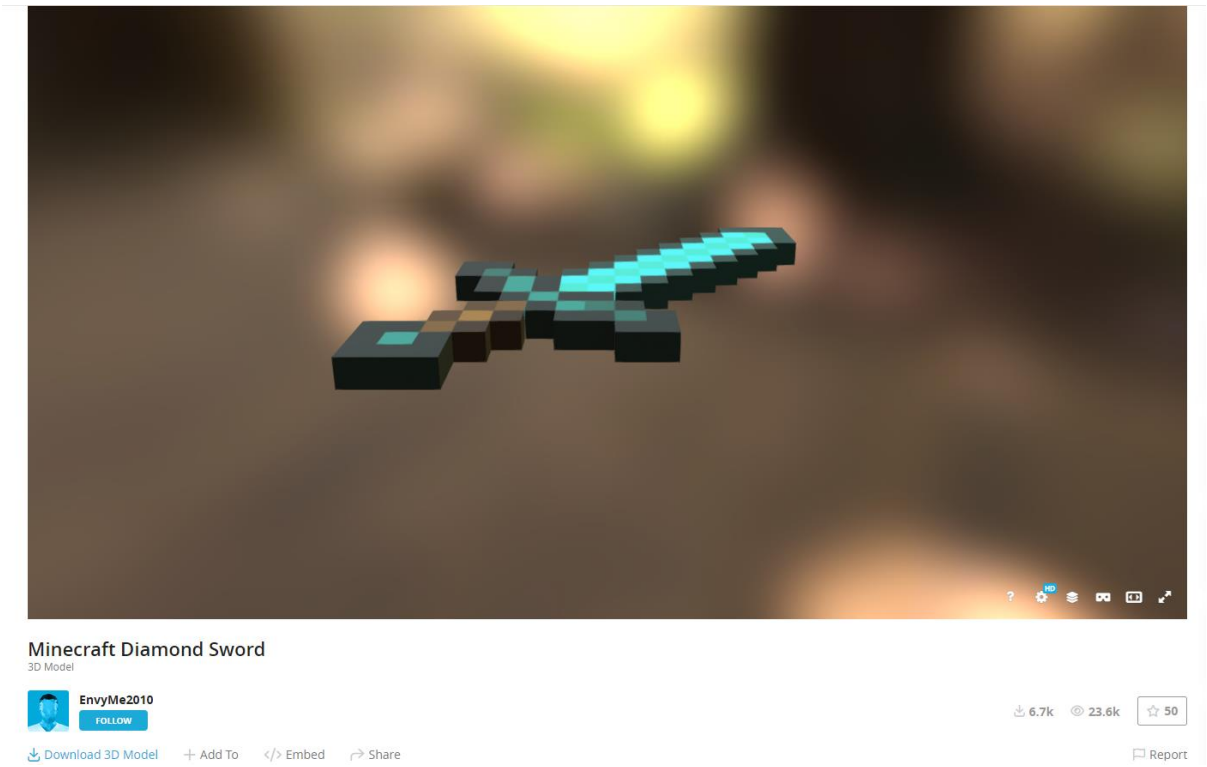


Adjust the collider height and radius according to your like and raise the collider above ground layer to avoid falling through the floor



Part 5: Creating a grabbable Object as a weapon to attack on enemies

Step 1: Download a sword asset



https://sketchfab.com/3d-models/minecraft-diamond-sword-567cb7974448493e871eaa6548aa3d80

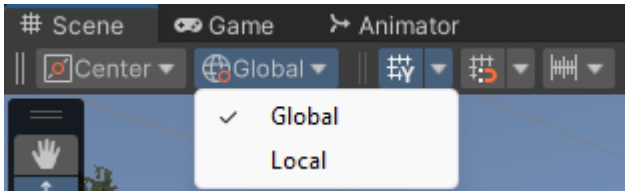
Step 2: Import the .fbx file into assets folder

Step 3: Create an empty game Object name it DiamondSword and add an empty child to it and name it AttachPoint, after that add the downloaded asset in the scene as a child of DiamondSword. The final arrangement should something like in the below image

Step 4: Adjusting the attach point

This will be the point where the player will grab the sword and we want it to be at the sword handle

In the scene tab instead of global select local axis



Select the attach point game object

X-axis (Red arrow) should point to the direction whose surface we desire to face right

Y-axis (Green arrow) should point to the direction whose surface we desire to face up

Z-axis (Blue arrow) should point to the direction whose surface we desire to face to the direction opposite to us, for a sword we want its tip to point away from us so we adjust the blue arrow towards the tip

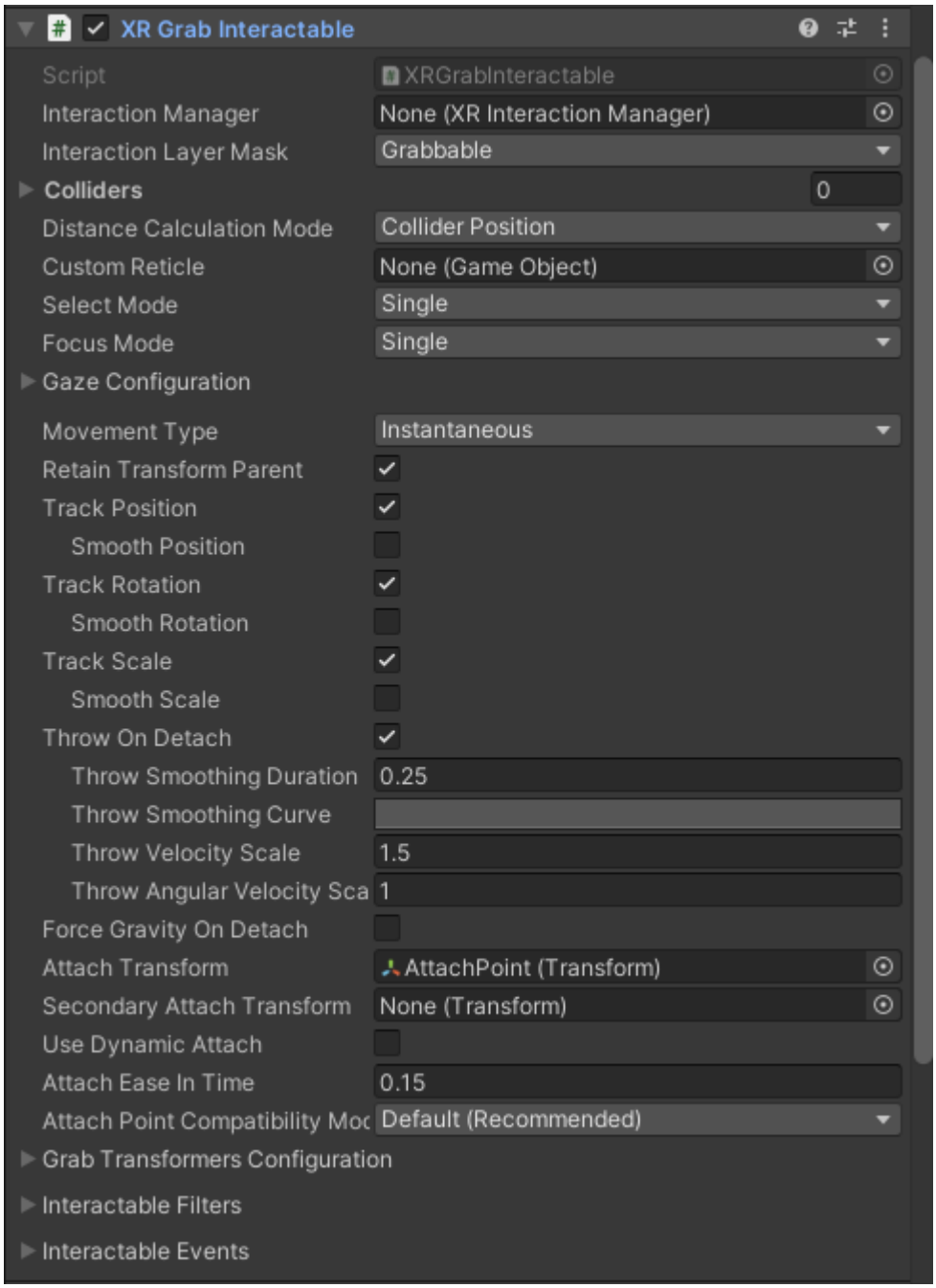
After that's done we want the player to grab the sword from its handle and not in middle so we position the attach point to the handle of the sword

Final setup should look something like this

Ignore the sound icon, see the direction and position of attach point

Step 5: To the DiamondSword game object add component XR Grab Interactable

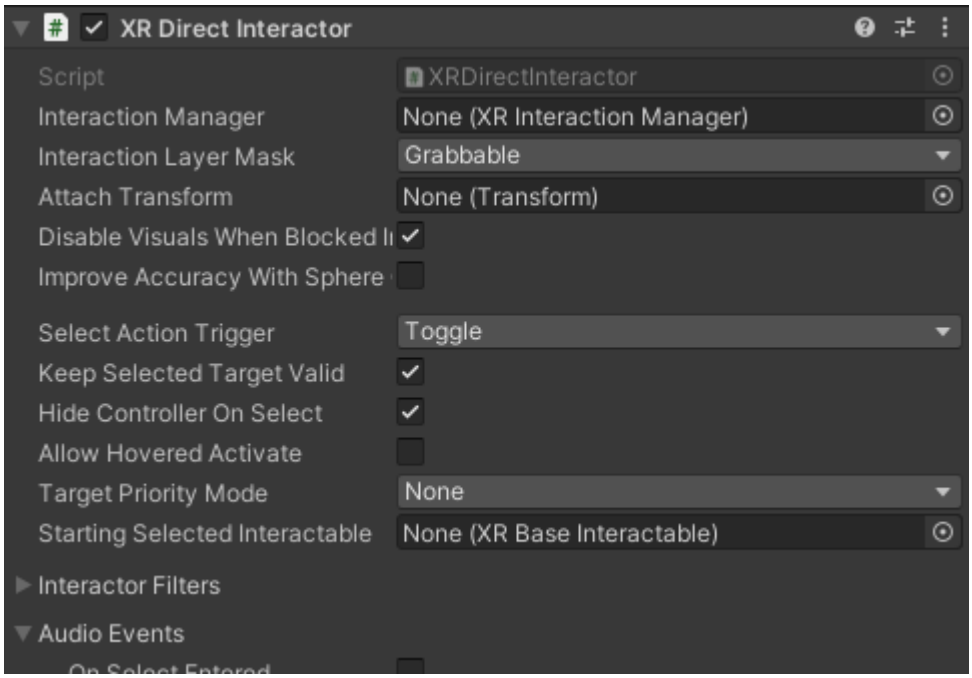
Scroll and find **Attach Transform** field and drag and drop the attach point we created in it



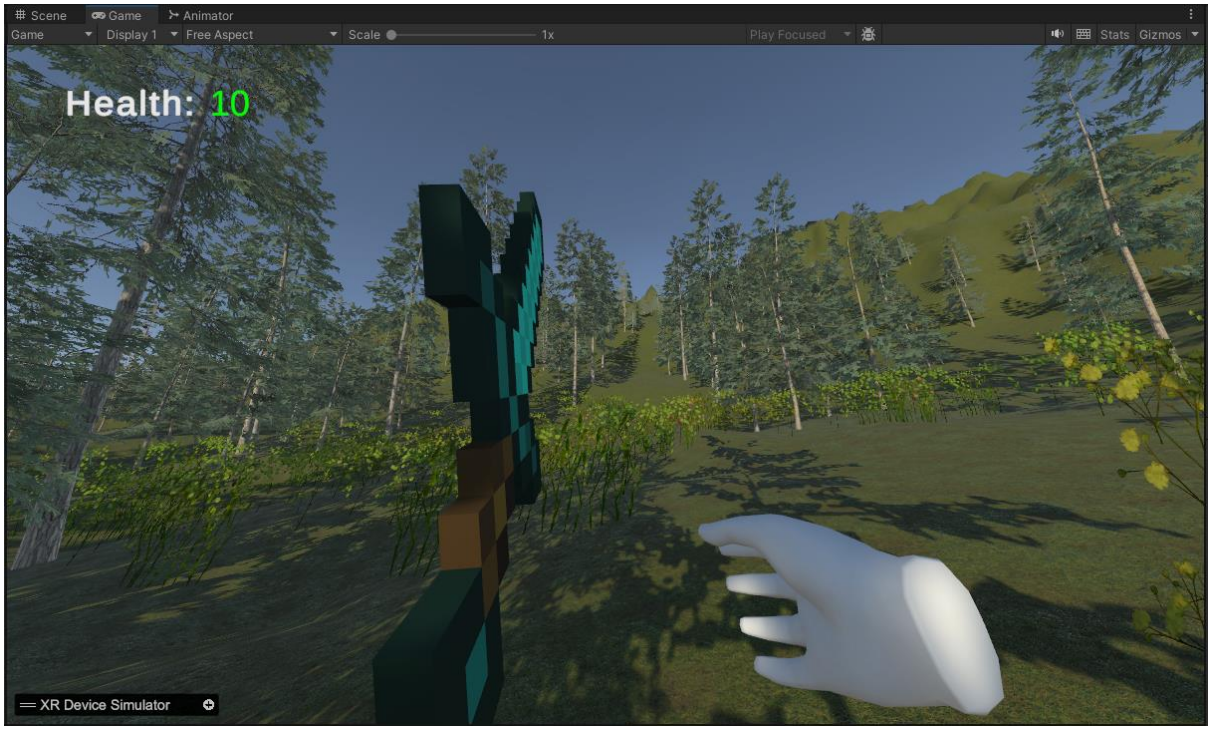
Step 6 (Optional): Disabling rendering of hands after grabbing onto the sword for better visibility

Go to the XR rig game object -> Camera Offset -> Left Controller -> Direct Interactor Component

And check Hide Controller on Select checkbox as well as Choose Select Action Trigger as Toggle instead of State Change so that the user has to only press the grab button once and the sword will remain in hands



Now the player can grab the sword at its handle



Task 7: Write VR Interaction Scripts

Part 1: Creating Player Health and Healing System

Step 1: PlayerHealth.cs Script attach to EventManager Script

```
Unity Script (1 asset reference) | 7 references
public class PlayerHealth : MonoBehaviour

{

    public TextMeshProUGUI playerHealth;
    public static int health = 10;
    public static int playerScore = 0;
    private bool onDeathScreen = false;
    public GameObject playerRig;
    public GameObject DeathScreenPanel;
    public TextMeshProUGUI ScoreCard;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        DeathScreenPanel.SetActive(false);
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        ScoreCard.text = playerScore.ToString();
        playerHealth.text = health.ToString();
        if(health < 6)
        {
            playerHealth.color = Color.yellow;

        }
        else if (health <= 2)
        {
            playerHealth.color = Color.red;
        }
        else
        {
            playerHealth.color = Color.green;

        }
        if (health <= 0)
        {
            //SceneManager.LoadScene("GameOver");
            //UnityEngine.Debug.Log("GAME OVER!!!");
            if (!onDeathScreen)
            {
                DeathScreen();

            }
        }
    }


    1 reference
    void DeathScreen()
    {
        onDeathScreen = true;
        DeathScreenPanel.SetActive(true);
    }

    0 references
    public void Respwawn()
    {
        DeathScreenPanel.SetActive(false);
        health = 10;
        onDeathScreen=false;
        playerScore = 0;
        ScoreCard.text = playerScore.ToString();
        playerRig.transform.position = new Vector3(348.940002f,-5.67999983f,64.211998f);

    }
}
```

Make sure to provide appropriate Game Objects to the script by creating them and dragging and dropping them like Death Screen as Canvas which has respawn button and player score

Functions of this script

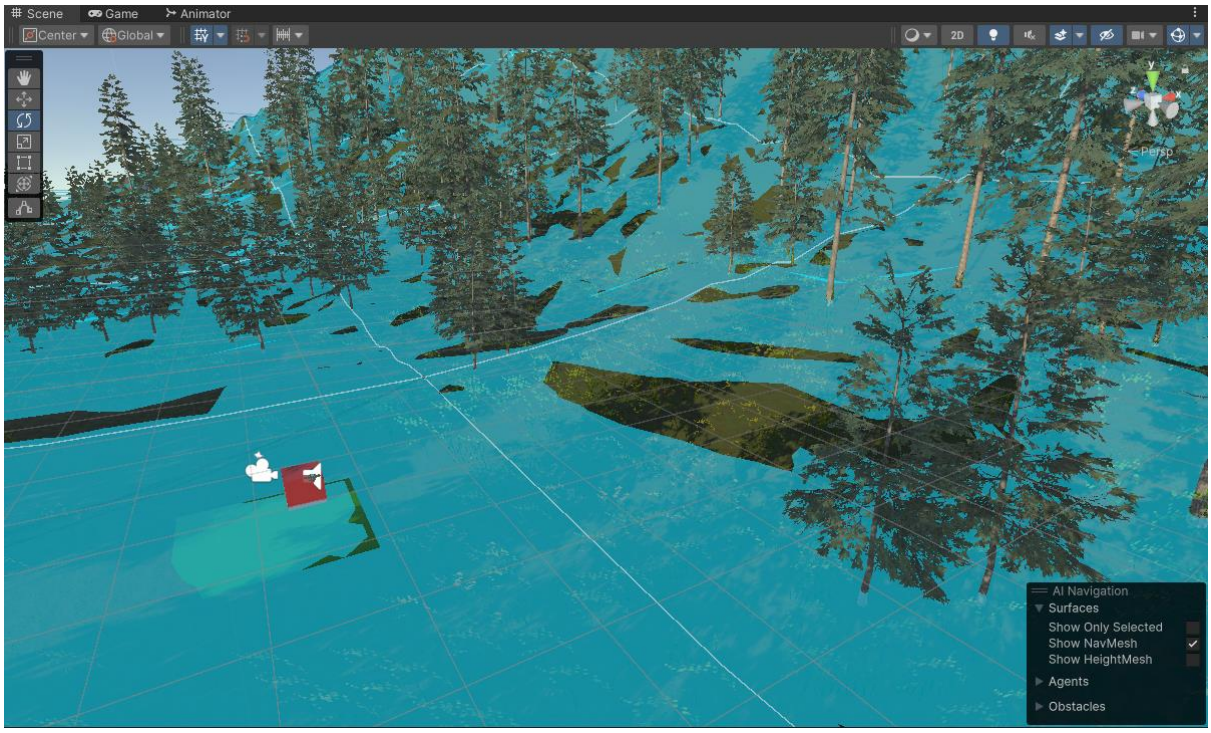
- Initialize Player health
- Show Player Death screen upon player death
- Respawn Player safely
- Track Score and reset upon death
- Alert Player about low health

Part 2: Zombie Controller and Navigation system

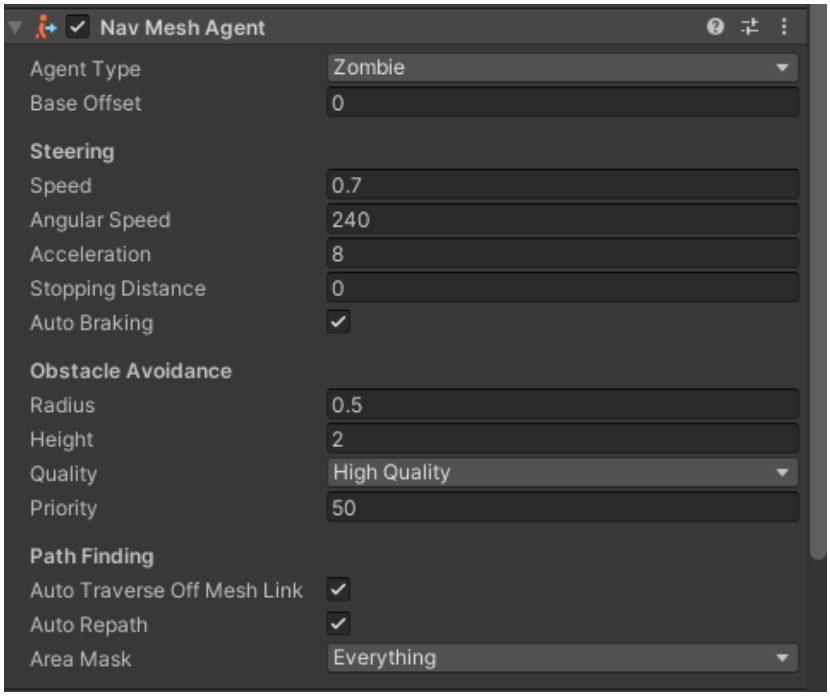
For navigation I will use Unity build in AI System

Step 1: Install AI Navigation from Package manager

Step 2: Apply NavMeshSurface to the terrain and bake it



Step 3: To the Zombie Prefab assign NavMesh Agent and give it appropriate speed



Step 4: Add the following scripts to the Zombie prefab

Capsule Collider

Rigid Body

ZombieController

Assembly-CSharp ZombieController

```
using System;
using UnityEditor.SearchService;
using UnityEngine;
using UnityEngine.AI;
using System.Collections;
using UnityEngine.SceneManagement;
using System.Runtime.CompilerServices;
using System.Collections.Generic;

Unity Script (1 asset reference) | 0 references
public class ZombieController : MonoBehaviour
{
    public Transform player;
    private NavMeshAgent agent;
    private float stoppingDistance = 1.0f;
    private Animator animator;
    public int ZombieDamage = 1;
    private static readonly int WalkTrigger = Animator.StringToHash("startwalking");
    private static readonly int AttackTrigger = Animator.StringToHash("t-pose");
    private static readonly int SitTrigger = Animator.StringToHash("idle");
    private static readonly int PushUpTrigger = Animator.StringToHash("pushup");
    private bool isAttacking = false;
    private bool onCooldown = false;

    private Coroutine healthReductionCoroutine;

    //public AudioClip spawnSound;
    private AudioSource audioSource;
    public List<AudioClip> zombieSpawn;
    public AudioClip damageSound;

    private string animationState = "s";
    public bool isStopped = false;

    private bool hasStartedMoving = false;

    Unity Message | 0 references
    void Start()
    {
        audioSource = GetComponent<AudioSource>();

        audioSource.PlayOneShot(SoundCentral.PlayRandomSound(zombieSpawn));


        //Debug.Log("Game started");
        agent = GetComponent<NavMeshAgent>();
        //agent.stoppingDistance = stoppingDistance;
        animator = GetComponent<Animator>();
        //TriggerSitAnimation();
    }
```

Assembly-CSharp ZombieController

```
    }

    Unity Message | 0 references
    void Update()
    {



        //if (!hasStartedMoving && agent.remainingDistance > 0)
        //{
        //    hasStartedMoving = true;
        //}
        agent.destination = player.position;

        if (PlayerHealth.health <= 0)
        {
            agent.isStopped = true;
            return;
        }
        //too far
        if (Vector3.Distance(agent.transform.position, player.position) >= 6.0)
        {
            agent.isStopped = true;
            isStopped = agent.isStopped;
            //Debug.Log("Too far");

            TriggerSitAnimation();
            //return;

        }
        //reached
        else if (Vector3.Distance(agent.transform.position, player.position) <= 1.2)
        {
            if (healthReductionCoroutine == null) // Check if coroutine is not already running
            {
                agent.isStopped = true;
                isStopped = agent.isStopped;
                //Debug.Log("Zombie has reached the player!");
                TriggerAttackAnimation();
                healthReductionCoroutine = StartCoroutine(ReducePlayerHealth()); // Start health reduction coroutine
            }
        }
        //chasing
        else
        {
            //Debug.Log("Chasing");

            agent.isStopped = false;
            TriggerWalkAnimation();
            isStopped = agent.isStopped;

            if (healthReductionCoroutine != null)
            {
                StopCoroutine(healthReductionCoroutine);
                healthReductionCoroutine = null;
            }
        }

    }
```

Assembly-CSharp

```
106     public void TriggerWalkAnimation()
107     {
108         if(animationState != "w")
109         {
110             animationState = "w";
111             animator.SetTrigger(WalkTrigger);
112         }
113     }
114
        1 reference
115     public void TriggerAttackAnimation()
116     {
117         if (animationState != "a")
118         {
119             animationState = "a";
120             animator.SetTrigger(AttackTrigger);
121         }
122     }
123
        1 reference
124     public void TriggerSitAnimation()
125     {
126         if (animationState != "s")
127         {
128             //Debug.Log("SIT TTRIFEWOAHDFAKLDJFHAJFEDHAEDJFHEDKFHJ");
129             animationState = "s";
130             animator.SetTrigger(SitTrigger);
131         }
132     }
133
        0 references
134     public void TriggerPushUpAnimation()
135     {
136         if (animationState != "p")
137         {
138             animationState = "p";
139             animator.SetTrigger(PushUpTrigger);
140         }
141     }
```

```
1 reference
private IEnumerator ReducePlayerHealth()
{

    onCooldown = true;


    PlayerHealth.health -= ZombieDamage;
    //Debug.Log("Player health: " + PlayerHealth.health);
    if(audioSource != null)
    {
        audioSource.PlayOneShot(damageSound);
        //audioSource.PlayOneShot(spawnSound);
    }
    else
    {
        Debug.Log("SPeakier gaaya");
    }

    if (PlayerHealth.health <= 0)
    {
        Debug.Log("Player is dead!");

    }

    // Wait for the cooldown duration
    yield return new WaitForSeconds(2);

    // End cooldown
    onCooldown = false;
}
}
```

This script is responsible for

Zombie Animation Control

Zombie Chasing and Attacking Player upon getting too close

Playing Appropriate Zombie sounds like taking damage, idle sounds, death sounds

ZombieHealth

```
Unity Script (1 asset reference) | 2 references
public class ZombieHealth : MonoBehaviour
{
    public float health = 10f;
    private AudioSource audioSource;
    public List<AudioClip> zombieHurt;
    public AudioClip zombieDeath;
    2 references
    public void TakeDamage(float damage)
    {
        audioSource = GetComponent<AudioSource>();
        //Destroy(gameObject);
        Debug.Log("Health left" + health.ToString());
        health -= damage;
        if (health > 0)
        {
            audioSource.PlayOneShot(SoundCentral.PlayRandomSound(zombieHurt));
        }

        Debug.Log("Zombie health: " + health);

        if (health <= 0)
        {
            //ZombieDeathSound.PlayDeathSound();
            //audioSource.PlayOneShot(zombieDeath);
            //// Handle zombie death (e.g., disable, destroy, etc.)

            //Destroy(gameObject);
            HandleZombieDeath();
        }
    }


    1 reference
    private void HandleZombieDeath()
    {
        PlayerHealth.playerScore += 1;
        //ZombieDeathSound.PlayDeathSound();
        audioSource.PlayOneShot(zombieDeath);


        GetComponent<Collider>().enabled = false;
        GetComponent<NavMeshAgent>().enabled = false;
        GetComponent<Animator>().enabled = false;


        StartCoroutine(DestroyZombieAfterDelay(1.5f));
    }

    1 reference
    private IEnumerator DestroyZombieAfterDelay(float delay)
    {
        yield return new WaitForSeconds(delay);
        Destroy(gameObject);
    }

}
```

This script is responsible for Controlling Zombie health and killing it when health reaches zero

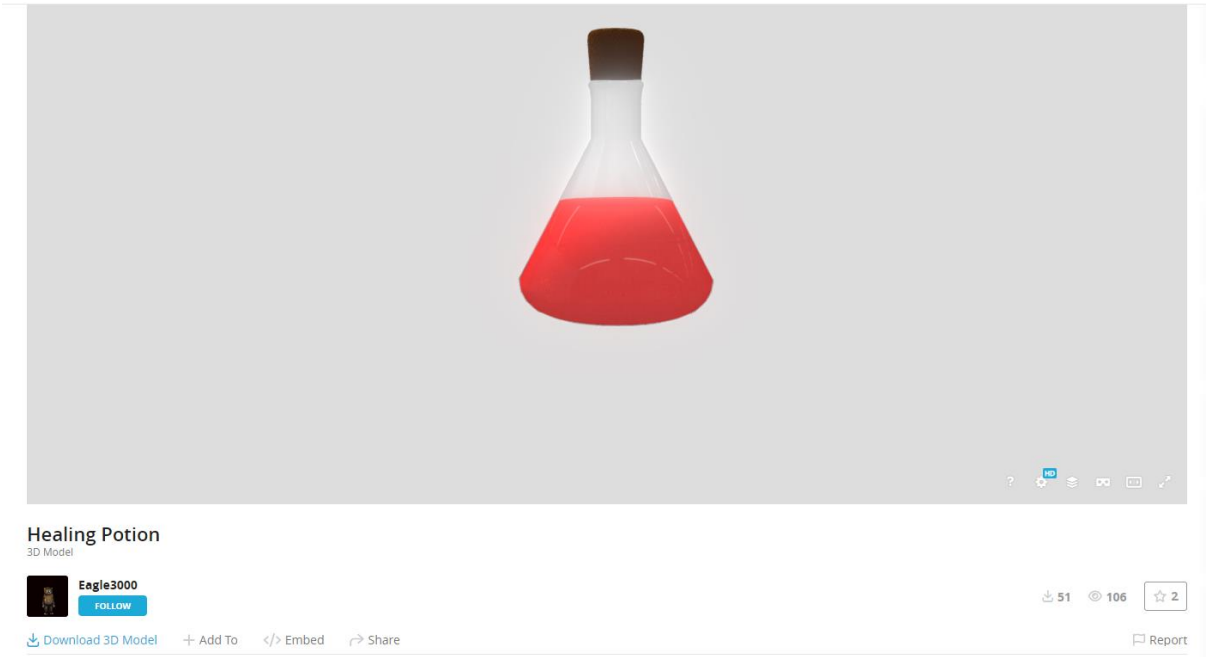
ZombieSpawner

```
Assembly-CSharp                                                    ZombieSpawner
using System.Collections;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class ZombieSpawner : MonoBehaviour
{
    public GameObject zombiePrefab;
    public Transform player;
    public Terrain terrain;
    public float spawnRadius = 10.0f;
    private int numberOfZombies = 1;

    Unity Message | 0 references
    private void Start()
    {
        StartCoroutine(SpawnZombiesCoroutine());
    }

    1 reference
    private IEnumerator SpawnZombiesCoroutine()
    {
        while (true)
        {
            SpawnZombies();
            yield return new WaitForSeconds(18);
        }
    }

    1 reference
    private void SpawnZombies()
    {
        if (numberOfZombies > 8)
        {
            numberOfZombies = 8;
        }

        for (int i = 0; i < numberOfZombies; i++)
        {
            Vector3 spawnPosition = player.position + Random.insideUnitSphere * spawnRadius;


            spawnPosition.y = 0; // Temporarily set y to 0 to avoid incorrect sampling
            spawnPosition.y = player.position.y;


            Instantiate(zombiePrefab, spawnPosition, Quaternion.identity);
        }

        Debug.Log("SPAWNED " + numberOfZombies.ToString() + "       !!!!!!!!!!!!!!!!!!!!!!");

        numberOfZombies = Mathf.Min(numberOfZombies * 2, 16);
    }
}
```

This script spawns' zombies in waves at random locations with increasing level of difficulty by exponentially increasing number zombies spawned in each wave

Part 3: Player Health and Healing system

Step 1: Add healing potion prefab to the asset folder



https://sketchfab.com/3d-models/healing-potion-35e43b8661544e8780384421c0a472e5

Step 2: Add the script to the potion prefab

```
Assembly-CSharp                                                    HealthPotion
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class HealthPotion : MonoBehaviour
{

    public Transform Player;
    public float detectionRadius = 2.5f;
    //public TextMeshProUGUI score;

    private Renderer Renderer;
    private bool isPassed = false;
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        Renderer = GetComponent<Renderer>();
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

        if (!isPassed && Vector3.Distance(Player.position, transform.position) <= detectionRadius)
        {

            isPassed = true;
            if (PlayerHealth.health <= 5)
            {

                PlayerHealth.health += 5;
            }
            else
            {
                PlayerHealth.health = 10;
            }
            Debug.Log("Destroying Potion");
            Destroy(gameObject, 0f);



        }
    }
}
```

This script is responsible for detecting if a player gets close to the potion, if so it heals the player

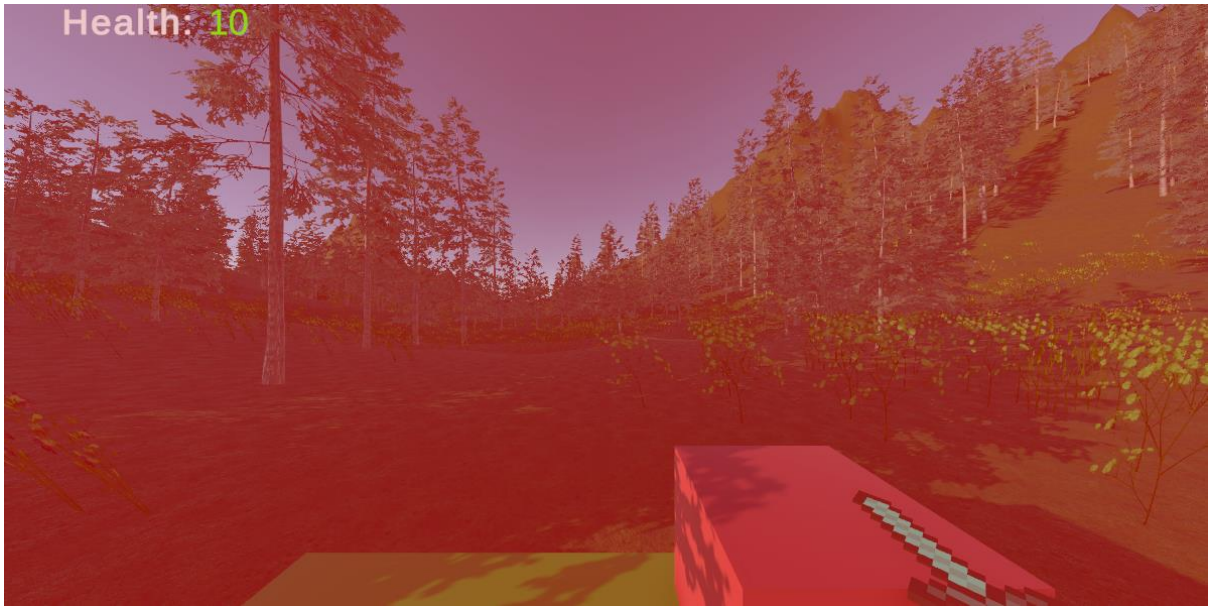
Step 3: Add the following script to the GameEventManager Component

```
Assembly-CSharp                                                    SpawnHealthPotions
using System.Collections;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class SpawnHealthPotions : MonoBehaviour
{
    public GameObject healthPotionPrefab;
    public Transform player;
    public Terrain terrain;
    public float spawnRadius = 5.0f;
    private int numberOfPotions = 1;

    Unity Message | 0 references
    private void Start()
    {
        StartCoroutine(SpawnHealthPotionsCoroutine());
    }

    1 reference
    private IEnumerator SpawnHealthPotionsCoroutine()
    {
        while (true)
        {
            SpawnHealthPotionsNow();
            yield return new WaitForSeconds(10);
        }
    }

    1 reference
    private void SpawnHealthPotionsNow()
    {
        if (numberOfPotions > 2)
        {
            numberOfPotions = 2;
        }

        for (int i = 0; i < numberOfPotions; i++)
        {
            Vector3 spawnPosition = player.position + Random.insideUnitSphere * spawnRadius;


            spawnPosition.y = player.position.y;

            Instantiate(healthPotionPrefab, spawnPosition, Quaternion.identity);
        }

        Debug.Log("SPAWNED " + numberOfPotions.ToString() + "       !!!!!!!!!!!!!!!!!!!!!!");

        numberOfPotions = Mathf.Min(numberOfPotions * 2, 16);
    }
}
```

This script is responsible for randomly spawning health potions near player so that player can keep heling while fighting enemies

Step 4: Giving Player Visual Feedback about health drops

The health colour changes from Green to yellow to red depending on player health level



Entire Screen will be flashed red when player takes damage

Part 4: Weapon Damage System

Step 1: Add a collider and rigid body to out sword

Step 2: Add the below script to it

```
Assembly-CSharp                                                    Sword
using UnityEngine;


using System.Collections.Generic;
Unity Script (1 asset reference) | 0 references
public class Sword : MonoBehaviour
{
    public float damage = 2f;
    public float speedThreshold = .00001f;
    private Rigidbody rb;
    private AudioSource audioSource;
    public List<AudioClip> playerAttackSounds;
    Unity Message | 0 references
    private void Start()
    {
        audioSource = GetComponent<AudioSource>();
        rb = GetComponent<Rigidbody>();
    }

    Unity Message | 0 references
    private void OnCollisionEnter(Collision collision)
    {
        ZombieHealth zombieHealth = collision.gameObject.GetComponent<ZombieHealth>();
        //Debug.Log(collision.gameObject + "zomibe mil gaya");
        //zombieHealth.TakeDamage(10);

        if (collision.gameObject.CompareTag("Zombie"))
        {
            Debug.Log("Sword hit the zombie!");
            zombieHealth.TakeDamage(damage);
            audioSource.PlayOneShot(SoundCentral.PlayRandomSound(playerAttackSounds));
            // Check the sword's speed or other conditions for damage
            Rigidbody swordRb = GetComponent<Rigidbody>();
            if (swordRb != null && swordRb.velocity.magnitude > speedThreshold)
            {
                // Apply damage to the zombie
                //ZombieHealth zombieHealth = collision.gameObject.GetComponent<ZombieHealth>();
                if (zombieHealth != null)
                {
                    zombieHealth.TakeDamage(5); // Assume you have a method to apply damage
                }
            }
        }
    }
}

```

This script is responsible for

Detecting Collision with the zombie

Reducing zombie health

Playing attack sounds upon successful hits

The sounds added for attacking were extracted from Minecraft Hit Sounds

Task 8: Creating a scoring mechanism

For our score system we will add a score for each zombie the player kills

The below script will implement that

```
1 reference
private void HandleZombieDeath()
{
    PlayerHealth.playerScore += 1;
    //ZombieDeathSound.PlayDeathSound();
    audioSource.PlayOneShot(zombieDeath);


    GetComponent<Collider>().enabled = false;
    GetComponent<NavMeshAgent>().enabled = false;
    GetComponent<Animator>().enabled = false;


    StartCoroutine(DestroyZombieAfterDelay(1.5f));
}
```

The script is already attached to the Zombie prefab under the class ZombieHealth

To reset the score we will implement the logic to do that in Respawn function

```
0 references
public void Respwawn()
{
    DeathScreenPanel.SetActive(false);
    health = 10;
    onDeathScreen=false;
    playerScore = 0;
    ScoreCard.text = playerScore.ToString();
    playerRig.transform.position = new Vector3(348.940002f,-5.67999983f,64.211998f);

}
}
```

After player presses on the respawn button their score will reset and they will be teleported to a safe spawn point

References

Terrain Asset By Unity - https://assetstore.unity.com/packages/3d/environments/landscapes/terrain-sample-asset-pack-145808

Grass Asset By ALP - https://assetstore.unity.com/packages/2d/textures-materials/nature/grass-flowers-pack-free-138810

Conifers Trees by forst - https://assetstore.unity.com/packages/3d/vegetation/trees/conifers-botd-142076

Ground Texture Pack by A dog's life software –

https://assetstore.unity.com/packages/2d/textures-materials/floors/outdoor-ground-textures-12555

Potion of healing - https://sketchfab.com/3d-models/healing-potion-35e43b8661544e8780384421c0a472e5

Diamond Sword - https://sketchfab.com/3d-models/minecraft-diamond-sword-567cb7974448493e871eaa6548aa3d80

Zombie Character - https://sketchfab.com/3d-models/zombie-4bd337e88ffb46909dc3021c5b620aa2

In game Audios - https://www.minecraft.net/en-us

Source Code GitHub: https://github.com/jeetavasare/VR-Assignment