VRTech.Lab4Report.Li

Name: Dazhi Li

RUID: 197007456

YouTube Link: https://www.youtube.com/watch?v=HSN8-P2Qv4Q

Google Drive Share (project file):
https://drive.google.com/file/d/1I1-OY7mvwVo9cTLBQABoTHXoWU6KvwQd/view?usp=sharing

1. Abstract

The scene I created in Lab 4 is called magic world, which is option A in Lab 4 assignment. Here I divide my scene into 3 parts. The left part has a camp and the right part has a lot of grass. The front part is 3 customized different tree used to test the wind zone spell. The whole view achieved by terrain is shown in figure 1.

Figure 1

2. Requirements

1) First-person controlled character with a wand visible on screen

Here I use the standard asset of unity to place a First Person Controller in my magic world. I set a child to it which is exactly the wand. Since we know if the parent is moving the child and grandchild will move with the parent or grand parent too. I set the wand exactly the right bottom of the player's camera. This will make the user feel that the wand is exactly hold by the user. The camera view of user is shown in **Figure 2-1**.

Figure 2-1

2) Passive and active sparking effects for the wand & Explosion by collision

First of all, I choose to use active sparking effect for the wand because the active sparking will make the user immerse into the magic world. Every time when the user is going to use the fire ball spell or light up spell the wand will automatically light up with purple particle system.

Secondly, when the fire ball is going to collide with other objects. The fire ball will play a particle system animation which shows explosion with a 3D explosion sound. You can see how fire ball works in my YouTube video and hear the audio.

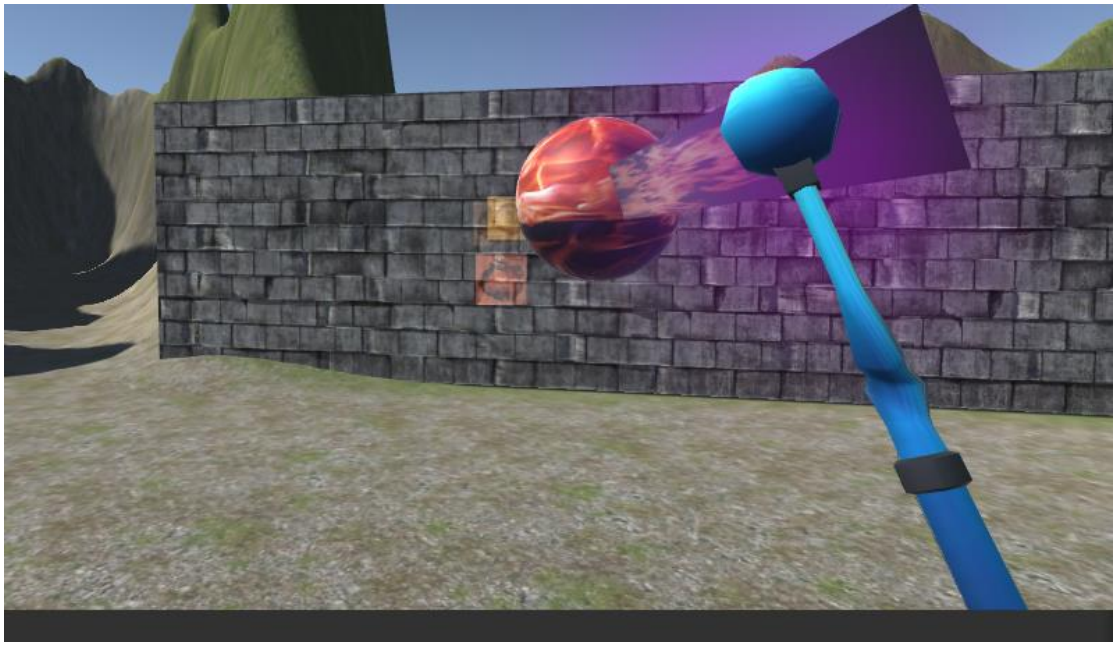

Figure 2-2

3) Landscape and forest

The landscape, in other word, the terrain is enormous. I simply follow the lab guidance to add some grass and trees and terrain textures to it.

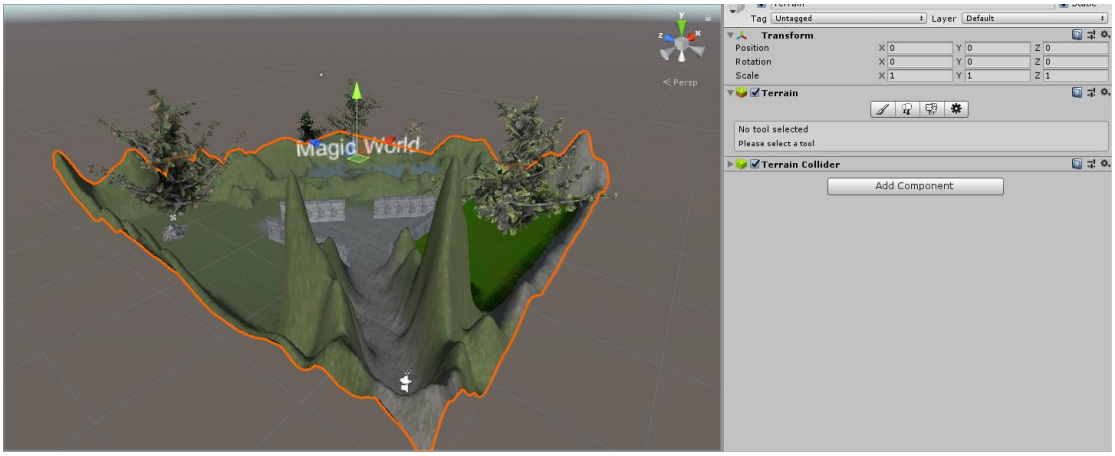

Figure 2-3

4) Ambient stereoscopic sounds

As we know the stereoscopic sound will be heard by the listener no matter how user turn his head, which is quite different from 3-D sound. As I mentioned above I created 3-D sound in my fire ball spell, you can feel difference between the ambient stereoscopic sound and 3-D sound. The ambient stereoscopic sound is shown in **Figure 2-4** which is kind of like BGM of the whole scene.

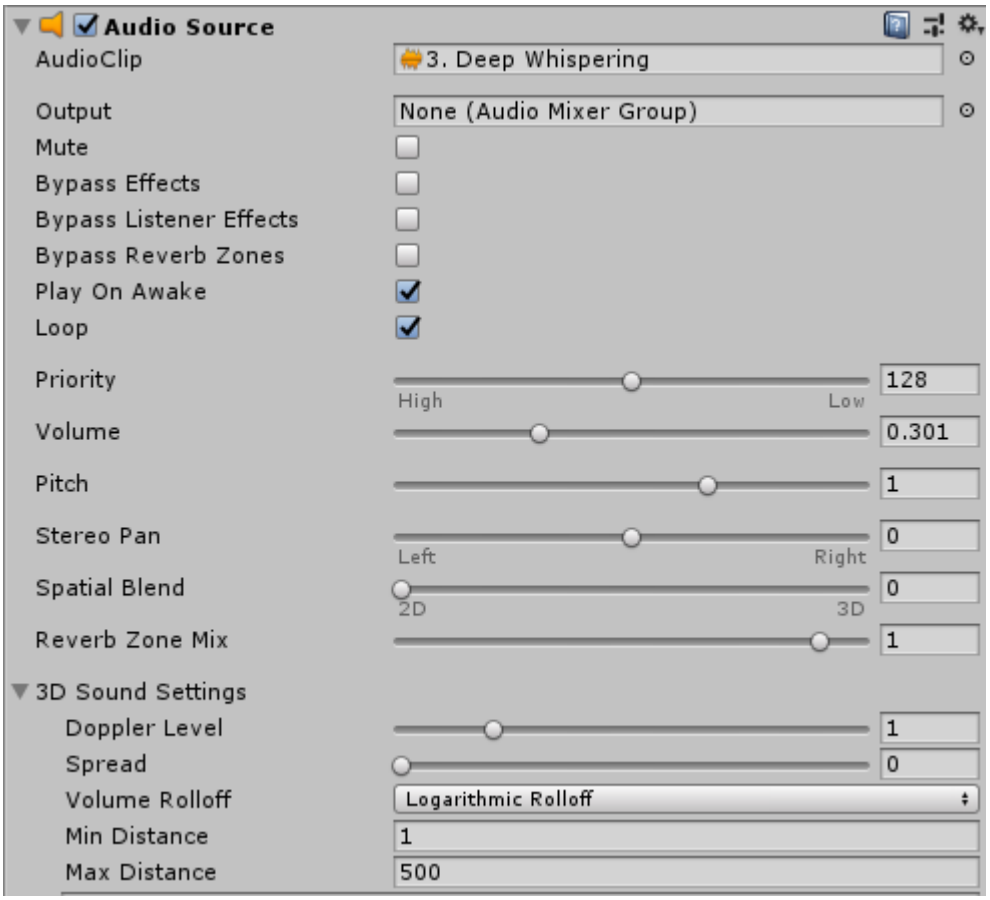

Figure 2-4

3. Advanced Work

1) Create 5 unique custom trees

There are totally 5 customized trees in my scene which is achieved by creating 3-D game objects named tree. By adding branches and leaves I created different trees in size/shape. By adding leaf texture or branch texture I created different trees in shaders. The first, second and third unique trees are shown in **Figure 3-1**. The forth tree is shown in **Figure 3-2**. The fifth tree is shown in **Figure 3-3**.

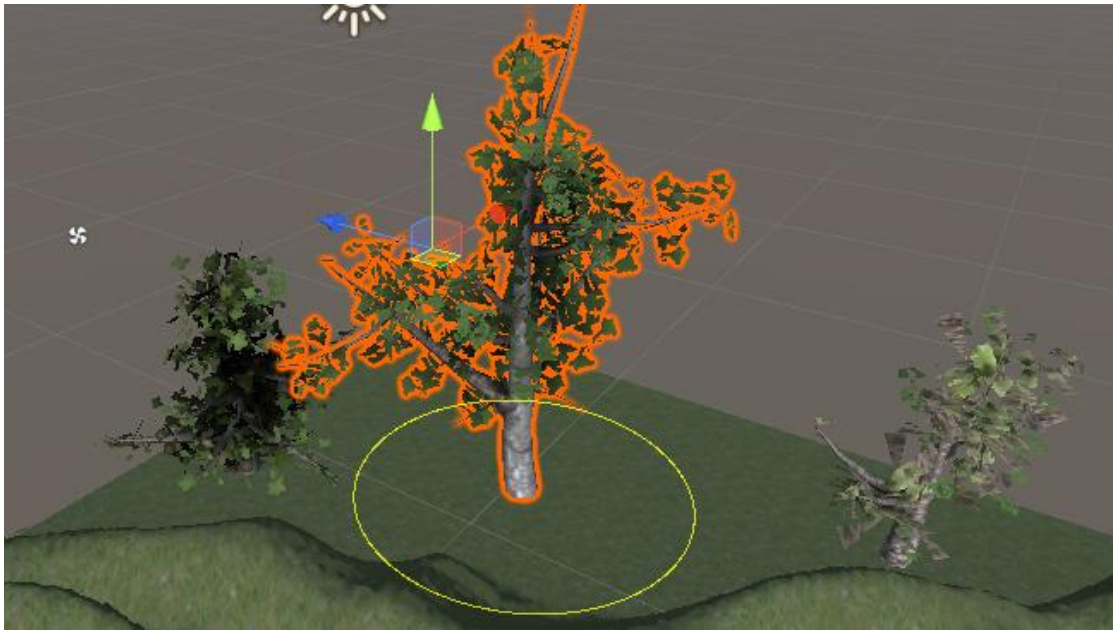

Figure 3-1

Figure 3-2

Figure 3-3

2) 5 unique spells

Fire ball is the most complete spell with audio source and collision detection particle system and spell trails. The fire ball is shown in **Figure 3-4**.

Figure 3-4

The second spell is creating ice fence. To make more fun in my project, I set the gravity of ice fence to be none. That means, any force could push the ice fence. The ice fence is shown in **Figure 3-5**.

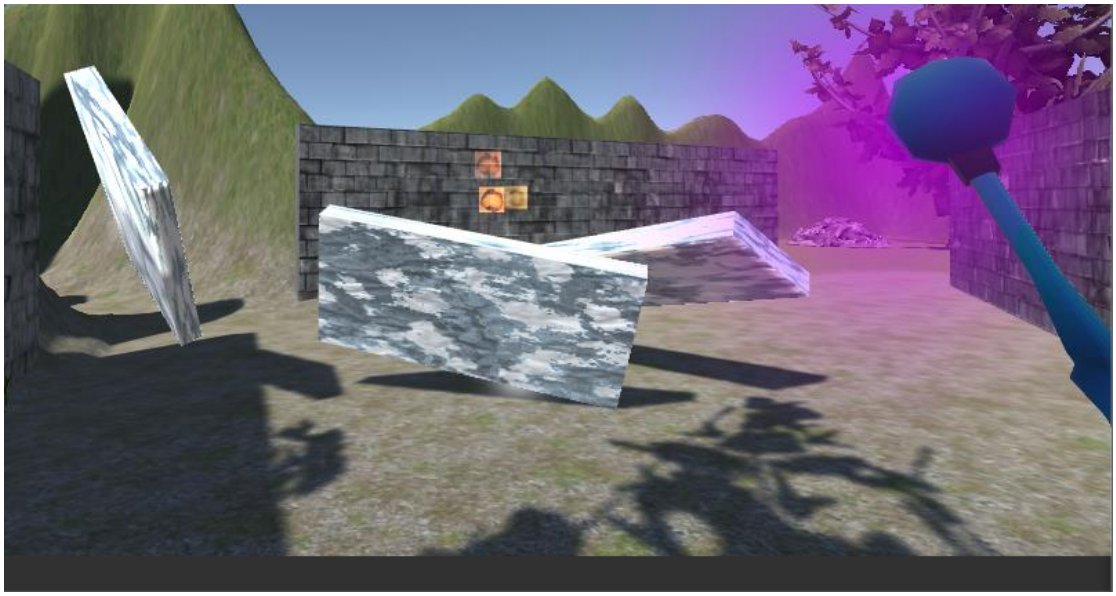

Figure 3-5

The third spell is creating bouncing ball from the sky which will fall down to the ground and bounce for a while. There is no fire rate in this function so that we can create as much as we want in a second. The bouncing balls are shown in **Figure 3-6**.

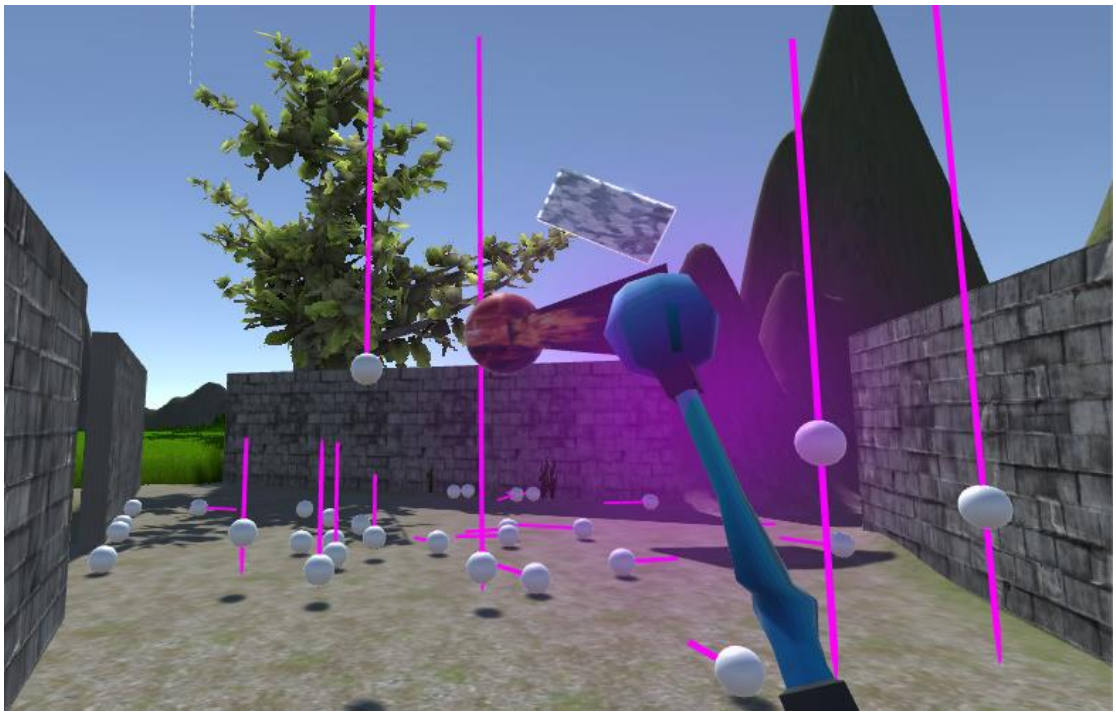

Figure 3-6

The forth spell is also another advanced requirement. **This spell is going to create dynamic wind zones that shifts around randomly**. Since the wind is invisible, we need other objects to observe how wind effects. From **Figure 3-7**, we can see a tree shrink because of the wind zone. The spell can add one wind zone to another. That means, the wind can be stronger by multiple wind zone together.

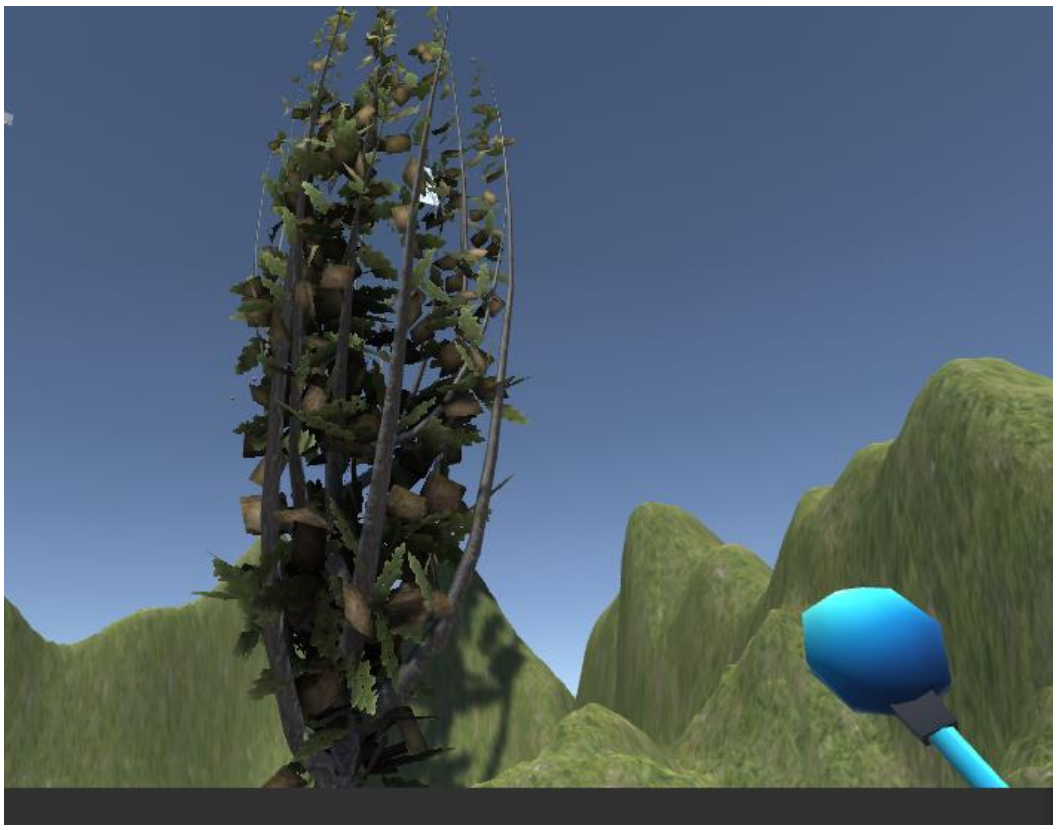

Figure 3-7

The last spell of my wizard wand is light the camp fire up or shut the fire down. As I face to the camp and click my left mouse button, the camp fire will automatically light up with smoke above or shut down. You can see my camp fire burning in **Figure 3-8**.

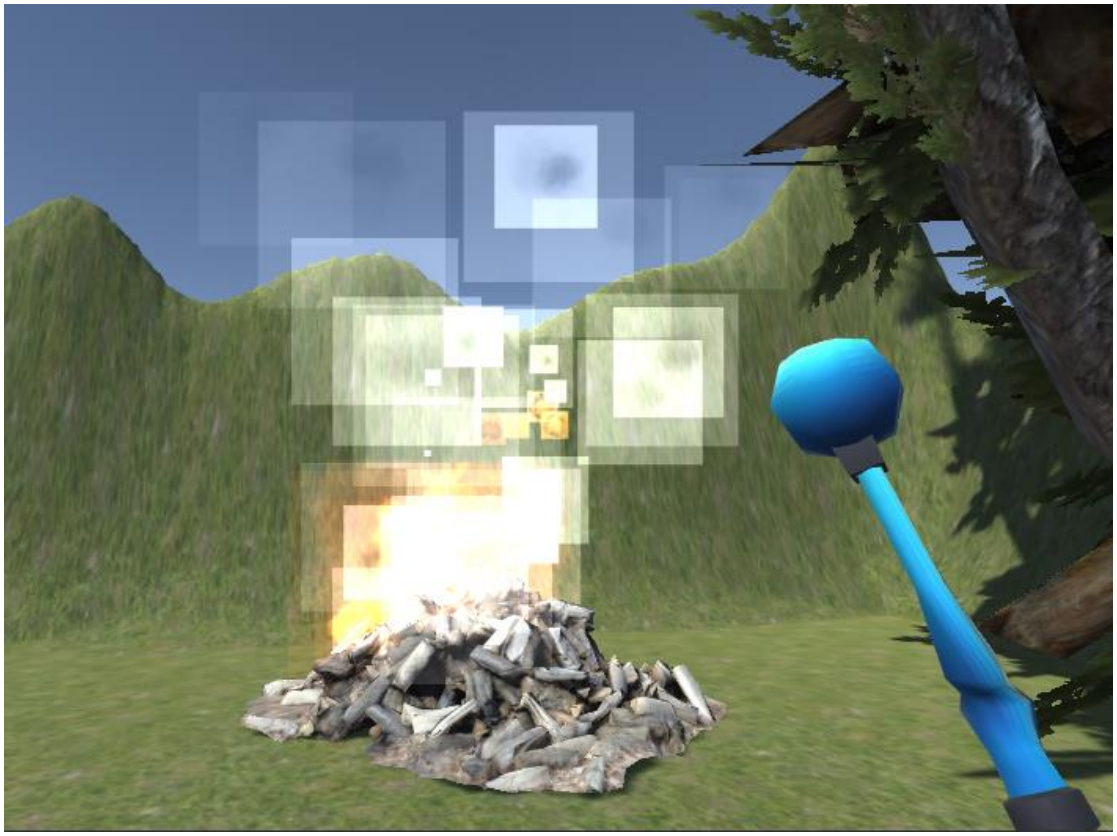

Figure 3-8

3) Spells must have trials

   There are two methods for adding a trail to a spell. The first method is adding a component of particle system to the spelled objects. For example, by adding a line particle system as the end of the object. The second method is what I have done in my project. I add a component named trial renderer to the spelled object, which will record every movement of the object and I can set the width of the trial and lifetime of the trial. Even I can pick up color/texture to the trial line. The trial can be seen in **figure 3-9**.

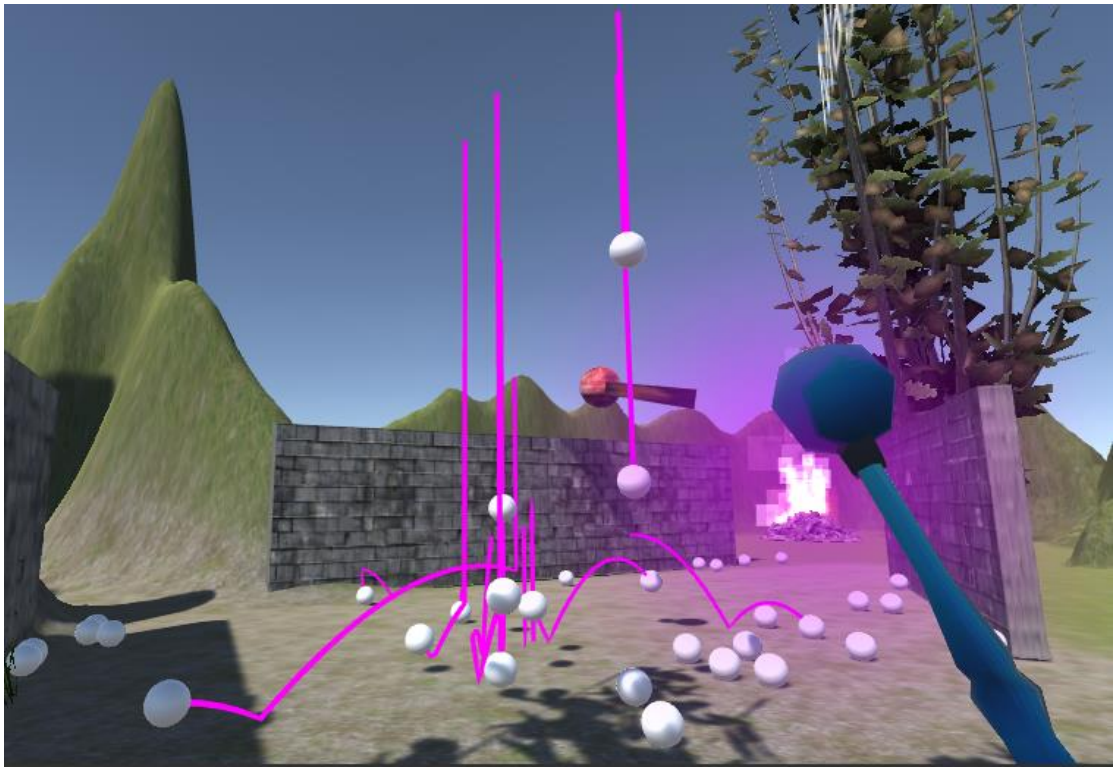

Figure 3-9

4) Stereoscopic sound effects on cast and on impact

I created two sounds in my fire ball spell. The first sound is shooting the fire ball and the second sound is fire ball explosion. The first one is stereoscopic sound and the second one is 3-D sound. You can hear how the stereoscopic sound perform in my project on my YouTube video.

4. Scripts

Wand_Spells

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class fire_ball : MonoBehaviour
{
    public float shootForce;
    public float fireRate;
    private float timeBeforeNextFire;
    [SerializeField]
    private GameObject fireball;
    [SerializeField]
    private GameObject icefence;
    [SerializeField]
    private Camera myCamera;
    [SerializeField]
```

```
        private GameObject fire_origin;
        [SerializeField]
        private GameObject ice_origin;
        [SerializeField]
        private GameObject bounce_ball;
        [SerializeField]
        private GameObject bounce_origin;
        [SerializeField]
        private WindZone wind_forece;
        [SerializeField]
        private GameObject wind_origin;
         // Update is called once per frame
         void Update()
        {
            if(Input.GetButtonDown("Fire1")&&Time.time>timeBeforeNextFire)
            {
                    GameObject clone=Instantiate(fireball,fire_origin.transform.position,myCamera.transform.rotation);
                    clone.SetActive(true);
                    Rigidbody cloneRB=clone.GetComponent<Rigidbody>();
                    cloneRB.AddForce(myCamera.transform.forward * shootForce);
                    timeBeforeNextFire=Time.time + fireRate;
              }
              if(Input.GetKeyDown(KeyCode.Z))
            {
                    GameObject clone=Instantiate(icefence,ice_origin.transform.position,myCamera.transform.rotation);
                    clone.SetActive(true);
              }
              if(Input.GetKeyDown(KeyCode.X))
            {
                    GameObject bb=Instantiate(bounce_ball,bounce_origin.transform.position,myCamera.transform.rotation);
                    bb.SetActive(true);
              }
              if(Input.GetKeyDown(KeyCode.C))
            {
                    WindZone wf=Instantiate(wind_forece,wind_origin.transform.position,myCamera.transform.rotation);
              }
        }
}
```

Ball_explosion

```
    public class ball_explosion : MonoBehaviour
    {
        [SerializeField]
        private ParticleSystem Explosion;
        [SerializeField]
```

```
        private GameObject FPS;
        [SerializeField]
        private GameObject Fireball;
        [SerializeField]
        private AudioSource sound;


        void OnCollisionEnter(Collision collision)
        {
                if (collision.gameObject==FPS)
                {
                }
                else
                {
                        ParticleSystem clone=Instantiate(Explosion,Fireball.transform.position,Fireball.transform.rotation);
                        AudioSource ex=Instantiate(sound,Fireball.transform.position,Fireball.transform.rotation);
                        clone.Play();
                        ex.Play();
                        Destroy(Fireball);
                }
        }
}
```

Fire_on

```
    public class fire_on : MonoBehaviour
    {
        [SerializeField]
        private ParticleSystem mParticles;
        [SerializeField]
        private ParticleSystem mParticles_second;
         // Start is called before the first frame update
    /*
         void Awake()
         {
                mParticles=GameObject.Find("fire").GetComponent<ParticleSystem>();
        }
    */
         // Update is called once per frame
         void OnMouseUp()
         {
              if(mParticles.isStopped)
              {
                        mParticles.Play();
                        mParticles_second.Play();
                }
                else
```

```
                {
                        mParticles.Stop();
                        mParticles_second.Stop();
                }
        }
}
```

Wand_light

```
    public class wand_light : MonoBehaviour
    {
        [SerializeField]
        private ParticleSystem mParticles;
         // Start is called before the first frame update
         // Update is called once per frame
         void Update()
         {
                if(Input.GetButtonDown("Fire1"))
                {
                        mParticles.Play();
                }
         }
}
```

5. Conclusion

   From this lab I learnt how to do unity terrain, trees and particle systems. Also, by using C# scripts will help us improve the interactive ability of unity to users. Finally, by adding audio source could make the whole scene more immersible.