MM-806 Assignment 2

1. Explanation of the work done

This assignment was developed with features such as navigation in the app including Panning(directional keys on laptop), zooming (touchpad/mouse wheel)), and moving forward when using navigation keys on laptop/keyboard.

In general, the movement in Unity can be done in two ways. Using the character controller component or using the rigid body component. And both definitely have their advantages depending on what we want to do.Now the main problem with creating a universal character controller that works for everyone is that what we want our character controller to do completely depends on our application.

Of course everyone wants movement. But there's also **gravity, jumping, drag, in-air movement, handling slopes, handling steps, sprinting, crouching, and interacting with physics objects** just to name a few. And some of these are easier to do with the character controller approach, while others are much easier with the rigid body approach. In fact, there's also quite a few things that you don't necessarily think about but are pretty important for the movement. Such as adding a ground check to know if the player is currently standing on the ground, and adding logic to make sure that the player doesn't get stuck on walls. There are different ways to create character controls but the simplest way of getting some really nice controls that can still handle movement, gravity, ground checking, jumping, in-air movement, steps, slopes, and that doesn't get stuck on walls is using the character controller component since much of this stuff is already built in. This way we can also easily add more things such as crouching or sprinting. The only real restriction is that the character controller doesn't interact with physics objects.

Before I started with the movement, I added a player with a camera that can look around. So the first thing that I did was to create an empty object. I named the object first person player and dragged it into the village scene. Next I added a character controller component. Search for character controller and hit enter. Of course, this wasn't very visible. So I created some graphics for so that we can easily see it within our scene. So I right-clicked on my player Object,Went to 3D object, and created a cylinder (Magenta colour). I then set the scale of this to 1.2, by 1.8, by 1.2 and removed the capsule collider. The reason for this is that the character controller itself also acts as a collider. Then I added a material to this and drag it onto my player. So now that I had my player in place, Then I added a main camera (I removed the unnecessary camera objects that were already present but were not in use).so that I can have a look around.

Now I have a main camera. And drag it under my player object. I reset the transform here and then I moved it up on the Y so that it's almost at the top, but still leave a tiny bit of space. The reason for this is that if I go ahead and jump into an object, I didn't want my camera to clip through. And also I wanted it to stay inside of our graphics object so that it's not visible to the camera itself.

And the first thing that I wanted to do was to create a player look script that was of course responsible for my ability to look around. Of course, everything is controlled with the **mouse**. And the mouse can be moved on two axis. **Mouse X and Mouse Y**. If I moved my mouse on the X, or from side to side,
I wanted to rotate my entire player around the y-axis. This allowed me to turn around. If I moved my mouse on the Y, or up and down, I didn't want to rotate my entire player. I just wanted to rotate my camera on the x-axis. This allowed me to look up and down. Now it was a good idea to limit the rotation of my camera to 180 degrees so that it doesn't flip to point behind the player. This is called clamping.

With that in place, I went ahead and used the following logic as shown in code screenshots below:

For Zoom

BasicCamera_Zoom.cs | mouse_look.cs | PlayerMovement.cs

Users > thisisutkarshpandey > Desktop > Desktop_Items > UofA > Fall_2021 > MM-806 _VR > Assignment_2 > Assignment_2 > Assets > BasicCamera_Zoom.cs

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

public class BasicCamera_Zoom : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if(Input.GetAxis("Mouse ScrollWheel") > 0)
        {
            GetComponent<Camera>().fieldOfView--;
        }
        if(Input.GetAxis("Mouse ScrollWheel") < 0)
        {
            GetComponent<Camera>().fieldOfView++;
        }

    }
}

```

For Panning and navigation

mouse_look.cs | PlayerMovement.cs

Users > thisisutkarshpandey > Desktop > Desktop_Items > UofA > Fall_2021 > MM-806 _VR > Assignment_2 > Assignment_2 > Assets > mouse_look.cs

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

public class mouse_look : MonoBehaviour
{
    public float mouseSensitivity = 100f;
    public Transform playerBody;
    float xRotation = 0f;

    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked;
    }

    // Update is called once per frame
    void Update()
    {
        float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
        float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;
        xRotation -= mouseY;


        transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
        xRotation = Mathf.Clamp(xRotation, -90f, 90f);

        playerBody.Rotate(Vector3.up * mouseX);

    }
}

```

⊗ 0 ⚠ 0

For Jump action and gravity and Physics

PlayerMovement.cs

Users > thisisutkarshpandey > Desktop > Desktop_Items > UofA > Fall_2021 > MM-806 _VR > Assignment_2 > Assignment_2 > Assets > PlayerMovement.cs

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

public class PlayerMovement : MonoBehaviour
{
    public CharacterController controller;
    public float speed = 12f;
    public float gravity = -9.81f;
    public float jumpHeight = 3f;
    Vector3 velocity;
    public Transform groundCheck;
    public float groundDistance = 0.4f;
    public LayerMask groundMask;

    bool isGrounded;

    // Update is called once per frame
    void Update()
    {
        isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance, groundMask);
        if(isGrounded && velocity.y < 0)
        {
            velocity.y = -2f;
        }

        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");

        Vector3 move= transform.right * x + transform.forward * z;
        controller.Move(move * speed * Time.deltaTime);

        if(Input.GetButtonDown("Jump") && isGrounded)
        {
            velocity.y = Mathf.Sqrt(jumpHeight * -2f * gravity);
        }

        velocity.y += gravity * Time.deltaTime;
        controller.Move(velocity * Time.deltaTime);

    }
}
```

⊗ 0 ⚠ 0

The Pan operation is performed when I simply move the mouse(without any buttons pressed), like hovering.

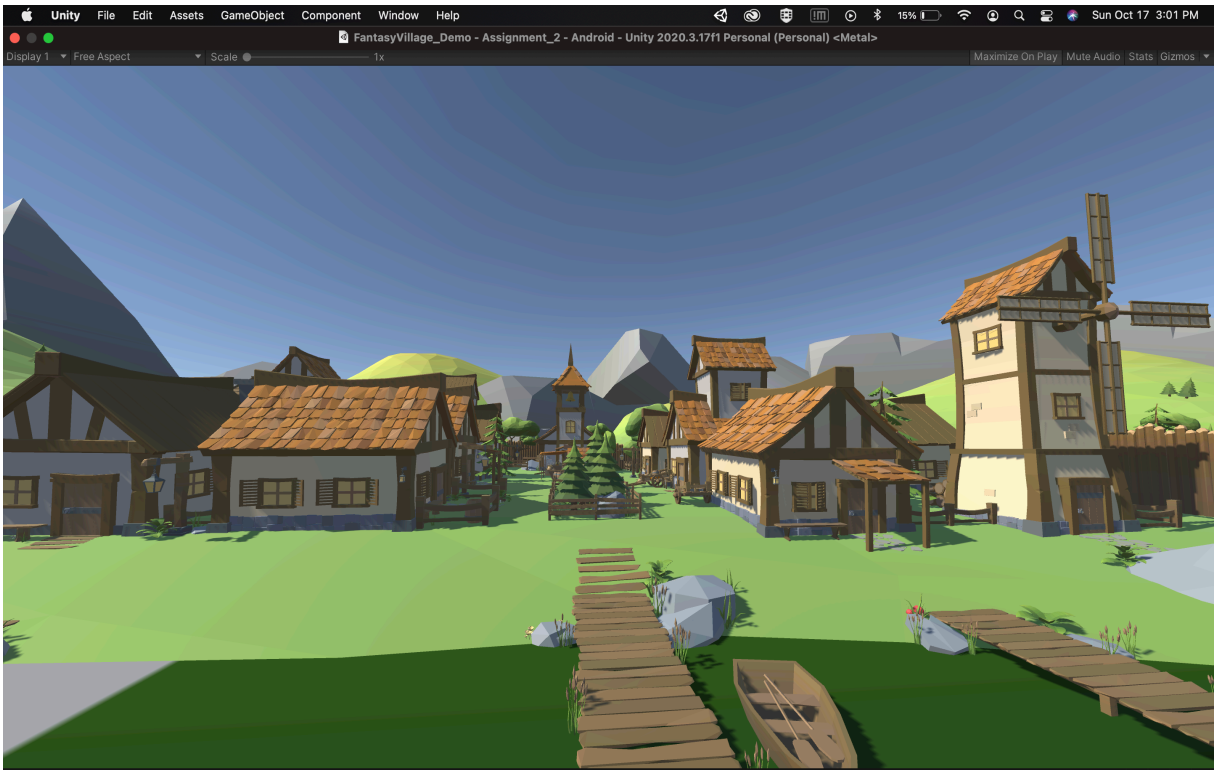
The navigation is performed when I use the directional keys on the keyboard like UP,DOWN, LEFT,RIGHT.

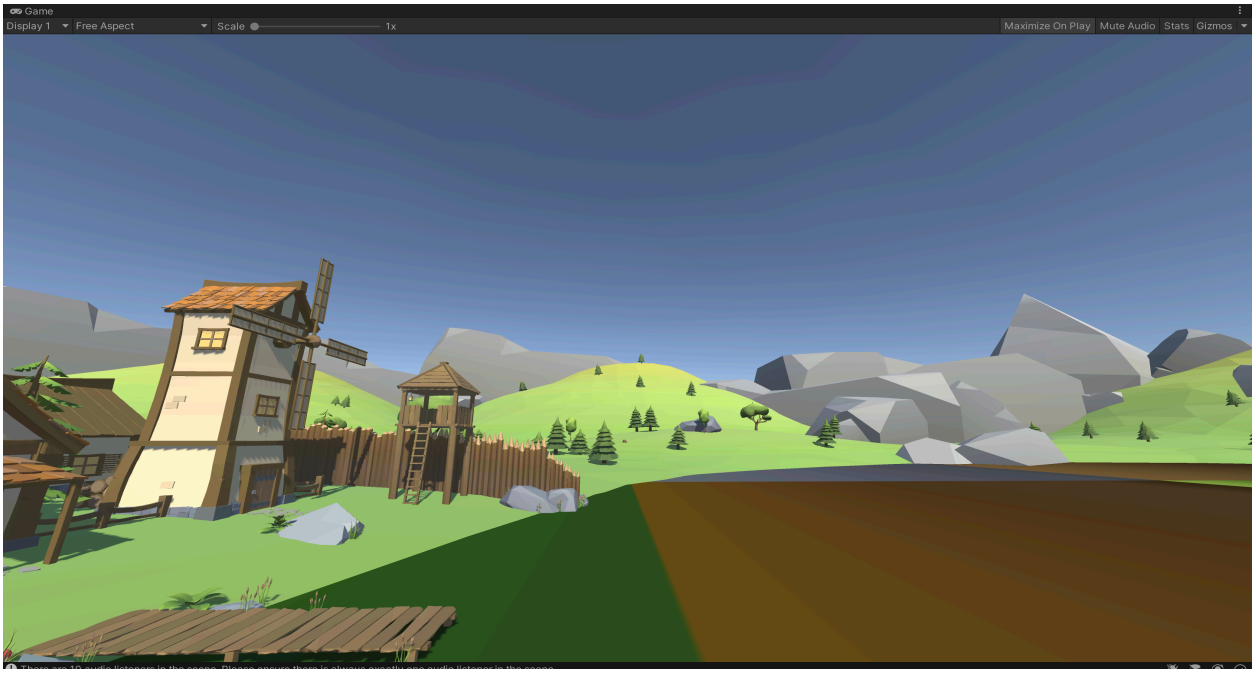
The Zoom in and Zoom out operation is performed when I use the mouse wheel to zoom in or zoom out.

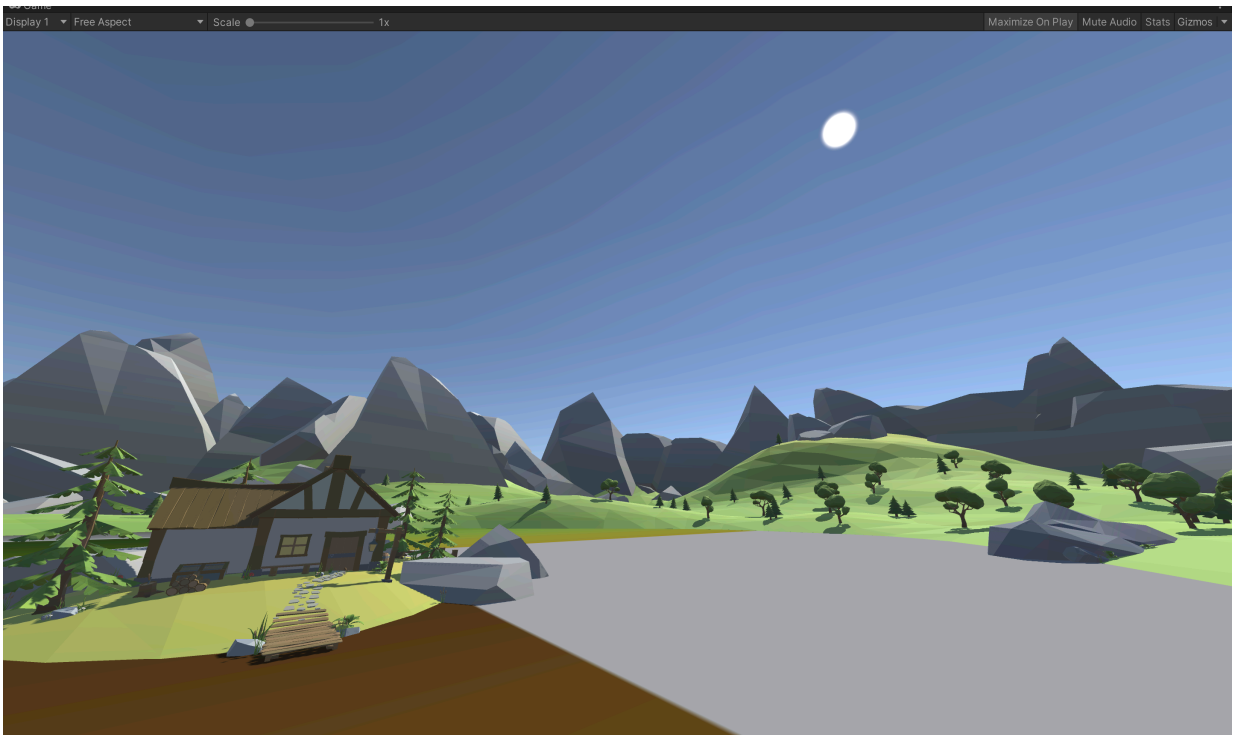
Pressing the SpaceBar Button acts Like a Jump action on the scene.

If we alter the Field Of View (FOV) of the Camera we can alter the zoom of the game scene and change the camera view point to the desired ratio

Below are the screenshots from the Game and the link to the running game on MacOS system.




Game
Display 1
Free Aspect
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos


Display 1
Free Aspect
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos

Here is the Link to the Video available on Google Drive:
https://drive.google.com/drive/folders/1jifd8LxG0dQ6jDw3IjH-gWcA8Z_VEUU0?usp=sharing

2. References