CM 3045

3D Graphics & Animation

Peer-graded Assignment: 2.503 First 3D Scene

Report

Abstract:

This report covers my “Peer-graded Assignment: 2.503 First 3D Scene” submission. Its contents and the decisions and thought process taken for their creation as well as an extra self-evaluation.

Introduction:

When thinking of a simple scene made with primitives and transforms, a church immediately becomes a great option as it is an enclosed space with limited lighting that can make use of primitives.

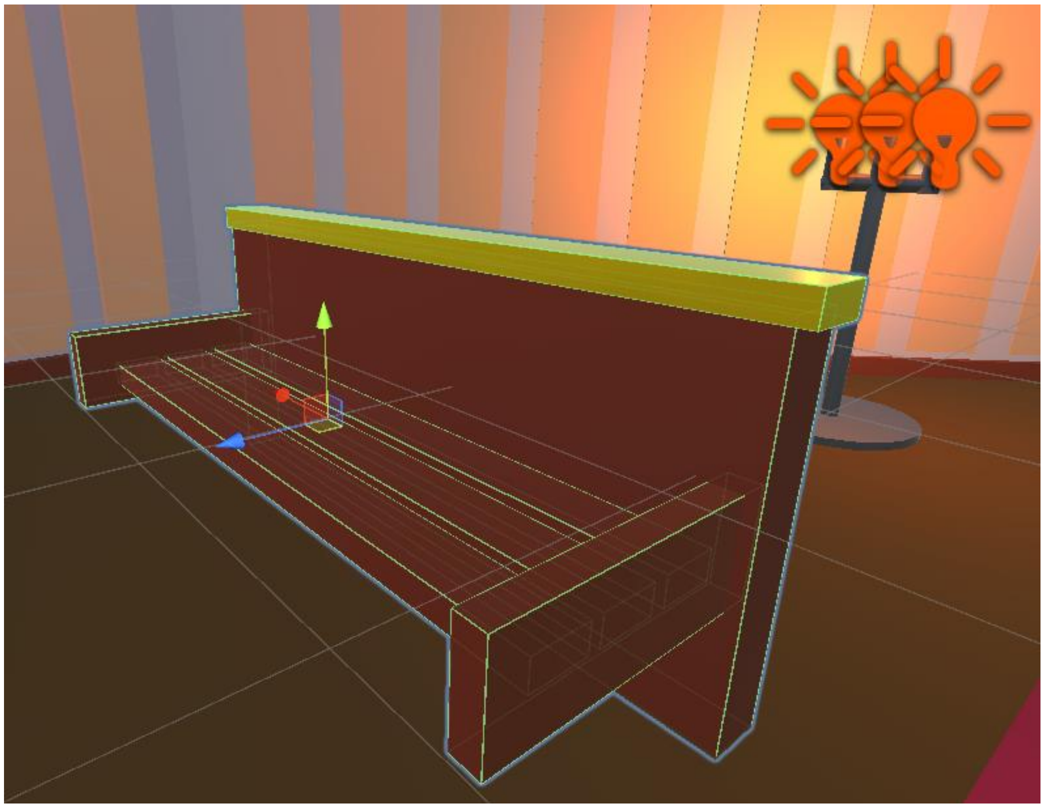
Task 1:

My first 3D scene, as the former paragraph suggests, is a church with 2 doors, a carpeted walkway through several benches, ending with a podium and a pastor’s booth. The white painted walls are paneled with birch wood, and a singular cross-shaped window overlooks the scene from behind the pastoral booth.

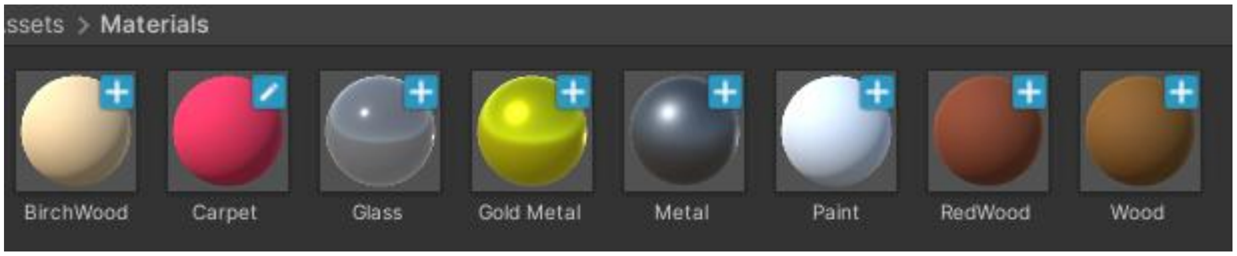
The scene features several light sources of varying types. From the entrance to the walkway, several point lights positioned at the top and floor illuminate the scene with warm colours as to strike a feeling of comfort and closeness rather than claustrophobia. The final steps before the podium hold a warm spotlight in order to define a clear place for a character in the scene to gaze at the pastoral booth that is elevated in order to impose a sense of grandeur. As the walls and ceiling expand to reveal the cross-shaped glass and its light beaming into the scene (directional), multiple wall-mounted point lights shine a strong dark blue both to symbolize heaven and to introduce colder motives into the scene – divinity is unreachable by mortals and all we can do is gaze upon it from below.

1. **3D Graphics & Animation techniques used:**
    a. Transformation (translation, scaling, rotation)

b. Combining primitives i.e.

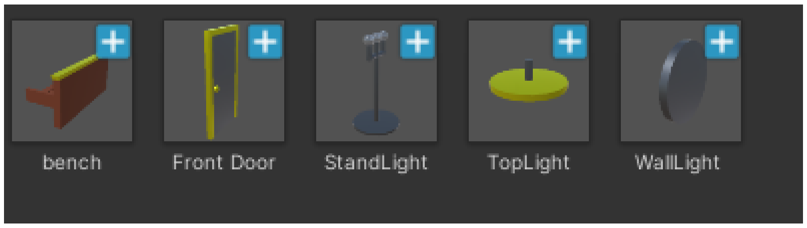


c. Custom Materials:

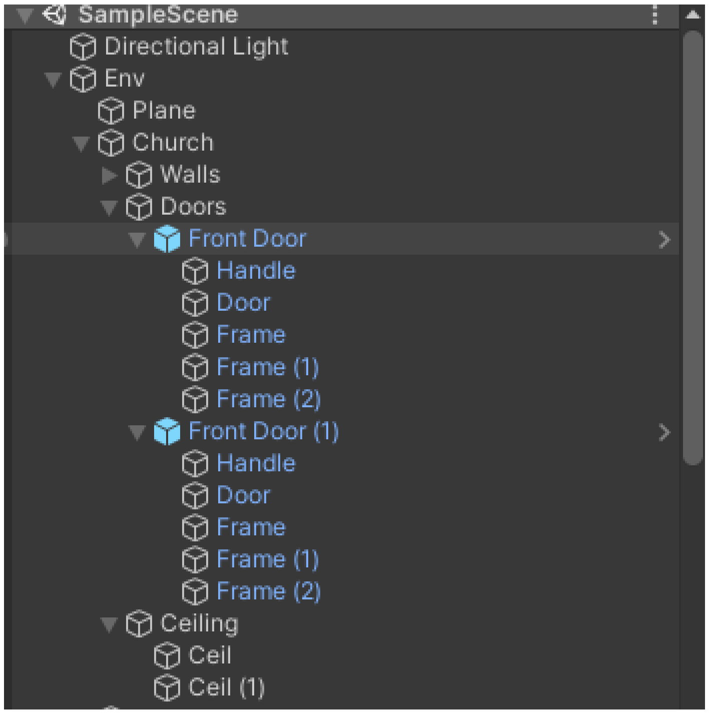


2. **Unity techniques used:**

a. Prefabs:

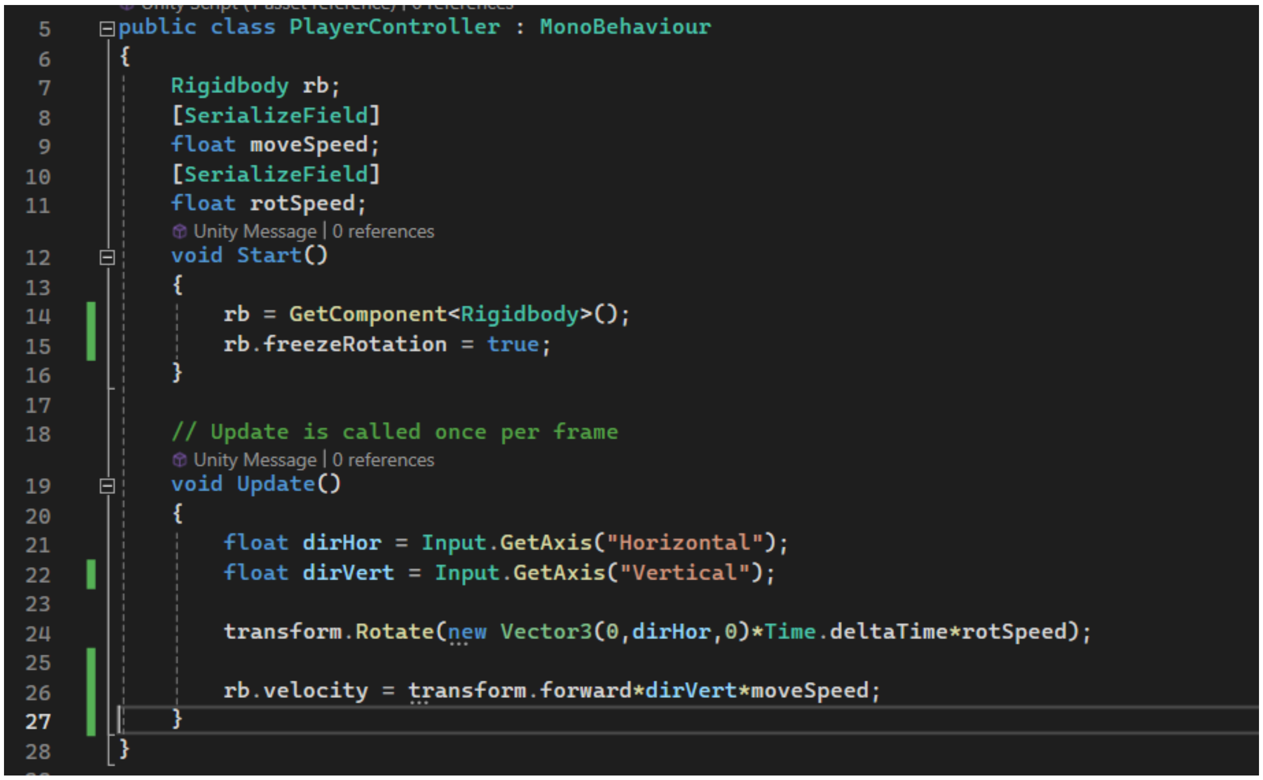


b. Grouping objects in hierarchy i.e. (small sample of the project hierarchy):


```
SampleScene
  Directional Light
  Env
    Plane
    Church
      Walls
      Doors
        Front Door
          Handle
          Door
          Frame
          Frame (1)
          Frame (2)
        Front Door (1)
          Handle
          Door
          Frame
          Frame (1)
          Frame (2)
      Ceiling
        Ceil
        Ceil (1)
```

Task 2 + Extension task/s:

1. **1 object that uses Input to move using transforms + Steering Behaviour:**
   I made a simple sphere and attached a rigid body to it. My code is my own as I have written many player controllers across my time as a hobbyist game dev and during the Video Games Development module. **Code:**

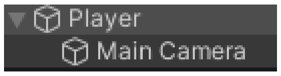


```
public class PlayerController : MonoBehaviour
{
    Rigidbody rb;
    [SerializeField]
    float moveSpeed;
    [SerializeField]
    float rotSpeed;
    Unity Message | 0 references
    void Start()
    {
        rb = GetComponent<Rigidbody>();
        rb.freezeRotation = true;
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        float dirHor = Input.GetAxis("Horizontal");
        float dirVert = Input.GetAxis("Vertical");

        transform.Rotate(new Vector3(0,dirHor,0)*Time.deltaTime*rotSpeed);

        rb.velocity = transform.forward*dirVert*moveSpeed;
    }
}
```

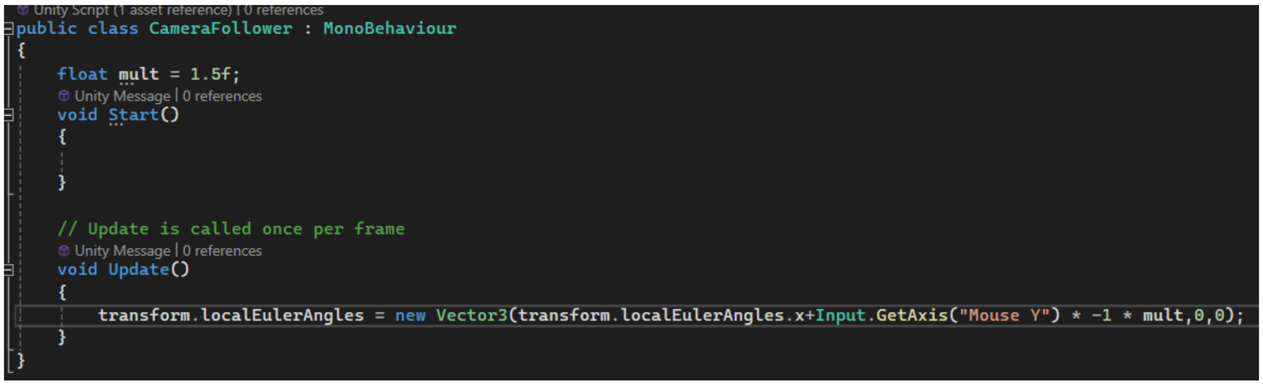
Code explanation:
3 variables are declared - the rigid body and 2 floating point numbers to store the desired movement and rotation speeds (serialized for easier access when testing). On the first frame of execution, the rigid body component is obtained from the player Game Object and its rotation is frozen. On every frame, the Horizontal and Vertical input axes are obtained (floating point number between -1 and 1; horizontal: -1 for A/left, 1 for D/right; vertical: -1 for D/down, 1 for W/up). Then, the transform is rotated using Euler angles with a new 3-dimensional vector - (0, horizontal input axis, 0) as to only rotate the transform on its Y axis (rotating side to side), multiplied by delta time (time in milliseconds from the previous to current frame) as to introduce motion and all of that is multiplied by the rotation speed in order to accelerate the movement. Then, the rigid body's velocity is set using the forward transform (a quick way of writing down (0,0,1) in order to only move the player on its Z axis (forward/back). This is multiplied by the vertical input axis to get either a forwards or backwards motion and by the move speed number to get a desired speed.

Using this code I have achieved a player controller that uses **steering behaviour** as it moves only forwards and backwards and its direction is controlled by steering.

2. **Implement a third person camera + camera control:**
As evident in the project, the main camera follows the player sphere from a third person perspective. This includes controlling translation and rotation, which I simply added by making it a child object of the player object (effectively inheriting its transform values):



Then, I was dissatisfied with the limited resulting viewport and wrote a small script that I attached to the main camera object that uses mouse Y to control X axis rotation:



```
Unity Script (1 asset reference) | 0 references
public class CameraFollower : MonoBehaviour
{
    float mult = 1.5f;
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        transform.localEulerAngles = new Vector3(transform.localEulerAngles.x+Input.GetAxis("Mouse Y") * -1 * mult,0,0);
    }
}
```

Code explanation:

I initialised a single floating point number to control the speed of the rotation. I used local Euler angles on each frame and set the rotation to a new Vector3. As Y rotation is already inherited, I am only concerned with the X axis which takes the current X rotation in Euler angles and adds to it the Mouse Y Axis, multiplies by -1 (to avoid what is known as inverted mouse movement) and then by the floating point number I initially made.

Conclusion and Evaluation:

I have covered all necessary points for both the basic requirements, the Task 1 & 2 Requirements, and covered 2 more expansion tasks – Steering controls and 3rd person camera. I have also added small improvements such as custom materials, ensured in-engine workflow was clean through proper grouping and inheritance and maintained clean code principles. I have made an effort to create a conceptual scene with meaningful choices in the architecture, lighting, colour and composition. Each asset used is my own and was created out of grouping 3D primitives.

For a beginning project I am satisfied with my work and I hope my effort is reflected through the final product. Some areas for improvement I can immediately notice are the limitations of using primitives when attempting to create complex, curved, or rotated shapes. Using custom meshes can easily alleviate that. The project runs less light sources than I originally created when running the build in WebGL and I need to learn how to optimise better. Despite my effort at making materials, this scene could be much better with custom textures with their own bump and/or displacement maps and it definitely lacks any visually appealing shaders and post-processing effects which I am eager to learn more about.