Virtual Worlds Assignment

Class : Virtual Worlds

Submission Date : 2024/04/14

Student ID : 2022742021

Affiliation : Kwangwoon University

Full name : Juho Kim

Major : Software Engineering

Abstract

This report provides a comprehensive walkthrough of the development process for the Virtual Worlds Programming Assigniment1 implemented in Unity 3D. First, the fulfillment of each requirement is discussed with descriptive code snippets. Then the report concludes by addressing the challenges encountered during development and outlining the limitations of the current code, which presents opportunities for future improvement.

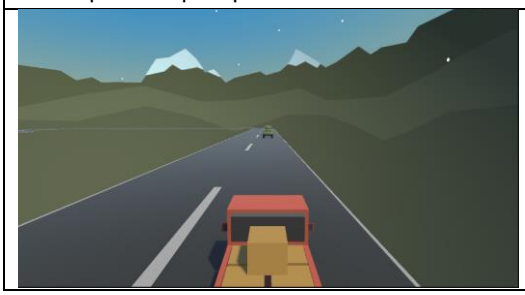
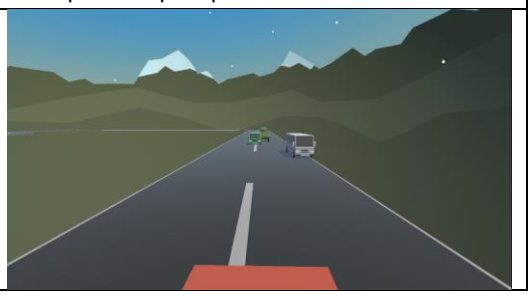
The game was developed applying the knowledge of vectors, quad, mesh and other concepts from Virtual Worlds. For more information, please visit my GitHub repository.

Requirements Fulfillment

Following table shows the assignment requirements from the slides along with completion status.

Easy1	Camera switcher	O
Easy 2	Revival	O
Medium 1	Oncoming vehicles	O
Medium 2	Road layout	O
Hard	Make a curved road mesh	O
Expert	Oncoming vehicle on curved road	△

Easy1 (Camera switcher)

Third person perspective	First person perspective
	

To keep an eye on the user's car, **FollowPlayer.cs** has been added to the Main Camera.

```
private Vector3 defaultOffset = new Vector3(0, 5, -7);
private Vector3 shiftedOffset = new Vector3(0, 5, -3);
```

In **FollowPlayer.cs**, defaultOffset and shiftedOffset each set the default position of the camera relative to the car at third-person and first-person perspective respectively.

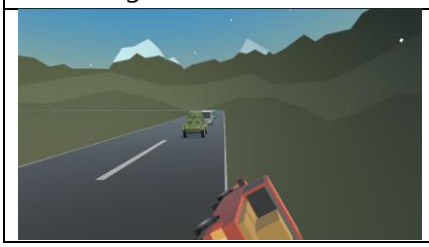
```
void LateUpdate()
{
    if (Input.GetKeyDown(KeyCode.LeftShift) || Input.GetKeyDown(KeyCode.RightShift))
    {
        shiftPressed = true;
    }
    else if (Input.GetKeyUp(KeyCode.LeftShift) || Input.GetKeyUp(KeyCode.RightShift))
    {
        shiftPressed = false;
    }

    transform.position = player.transform.position + (shiftPressed ? shiftedOffset : defaultOffset);
}
```

The car is set to third person perspective by default. If the user presses the shift key, **shiftPressed** becomes true and the camera moves to shiftedOffset, first person perspective.

.

Easy 2 (Revival)

Car falling	Car revived to starting position
	

If the car falls below the road, it is revived at the starting position as shown in the picture above.

To accomplish this, the position of the car is updated to its starting position if it falls below y=-10 in **PlayerController.cs**.

```
if (transform.position.y < -10)
{
    transform.position = new Vector3(127, 6f, 30);
    transform.rotation = Quaternion.Euler(Vector3.zero);
}
```

For the upcoming vehicles, the position of the vehicle is updated to its starting position if it falls below y=-10 in **MoveVehicles.cs**.

```
if (transform.position.y < -10)
{
    transform.position = new Vector3(127, 6f, 200);
    transform.rotation = Quaternion.Euler(0, 180, 0);
}
```

Medium 1 (Oncoming vehicles)

All upcoming vehicles are set to come towards the user from the other side at a constant speed in **MoveVehicles.cs** .

```
transform.Translate(Vector3.forward * Time.deltaTime * speed);
```

Medium 2 (Road layout)

The initial setting for the curved road was set as follows:

```
public float radius = 5f;
public float startAngle = 0f;
public float endAngle = 90f;
public int segments = 10;
```

The vertices were set using triangular function, cos and sin as stated below :

```
protected override void SetVertices()
{
    float angleStep = (endAngle - startAngle) / segments;

    for (int i = 0; i <= segments; i++)
    {
        float currentAngle = startAngle + (angleStep * i);
        float currentRadian = currentAngle * Mathf.Deg2Rad;

        float x = Mathf.Cos(currentRadian) * radius;
        float z = Mathf.Sin(currentRadian) * radius;

        Vector3 outerPoint = new Vector3(Mathf.Cos(currentRadian) * (radius + size / 2f), 0f, Mathf.Sin(currentRadian) * (radius + size / 2f));
        Vector3 innerPoint = new Vector3(Mathf.Cos(currentRadian) * (radius - size / 2f), 0f, Mathf.Sin(currentRadian) * (radius - size / 2f));

        vertices.Add(outerPoint);
        vertices.Add(innerPoint);
    }
}
```

Then the corresponding normal and UVs were set using iteration

```
protected override void SetNormals()
{
    for (int i = 0; i < vertices.Count; i++)
    {
        normals.Add(new Vector3(0, 1f, 0f));
    }
}
```

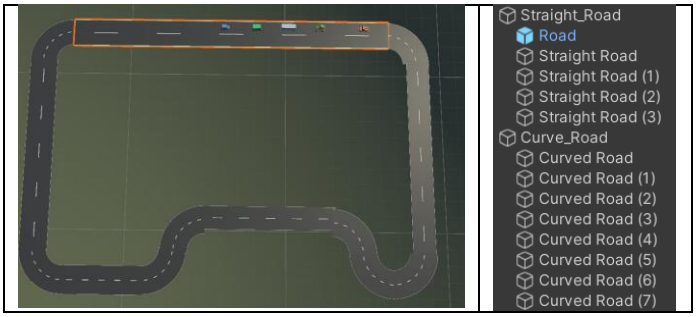
```
protected override void SetUV()
{
    for (int i = 0; i < vertices.Count / 2; i++)
    {
        uv.Add(new Vector2(i / (float)segments, 1));
        uv.Add(new Vector2(i / (float)segments, 0));
    }
}
```

Finally, the vertices created were looped through to create multiple triangles for forming the curved road.

```
protected override void SetTriangles()
{
    for (int i = 0; i < segments; i++)
    {
        int startIndex = i * 2;
        triangles.Add(startIndex);
        triangles.Add(startIndex + 1);
        triangles.Add(startIndex + 2);

        triangles.Add(startIndex + 2);
        triangles.Add(startIndex + 1);
        triangles.Add(startIndex + 3);
    }
}
```

Hard (Make a curved road mesh)



The track was created using 6 straight roads and 8 curved roads as shown above. The curved road made from the previous step were rotated to create different parts of the curve. Also, the straight road was replicated with the length adjusted for different parts of the track. Finally, all the roads have been classified into **Straight_Road** and **Curve_Road**.

Expert (Oncoming vehicle on curved road)

Attempts to translate the object on the curved road has been made as follows :

KakaoTalk_20240418_195028795.mp4

However, the upcoming object started translating once it reached the brink of falling from the curved road which made the assignment incomplete. Hence the following code was removed for clearing out any confusion.

Conclusion

Key challenges tackled

Some of the hardest parts were:

1. Creating two view perspective without the use of multiple camera

Initially, I started off by creating two cameras to switch to different cameras when user presses enter.

However this was not only made the code more complex but also added the burden of using an extra camera. Hence the code was refined by removing the camera and changing the position of one camera whenever user presses shift.

2. Creating the curved road

The curved road was the most difficult task of all because it involved the utilization of trigonometric functions for creating vertices using the for loop. After watching youtube tutorials on meshes, quads and Bezier curve, I was able to get a grasp of the mechanics for creating the curved road. And with endless trial and error, the curved road was created.

3. Creating road track

Creating the curved road was relatively easy compared to the curved road. However, aligning the mesh at run time and writing all the positions back again at program exit took a long time.

Limitations

1. Incoming vehicles unable to turn on curved road
2. Camera not turning when user goes through curve.

Future improvements

3. Drift Oncoming vehicle on curved road
4. Rotate camera when user goes through curved road

References

- Procedural Mesh tutorial
- Application of Bezier curve
- 2D Curve Introduction and Concepts