가상세계 **HW #1. Driving Cars**

제 출 일	**2024. 4. 16.(**화**)**	전 공	소프트웨어학부
과 목	가상세계	학 번	**2017203082**
		이 름	김찬진

목차

1. Fulfil table

Easy 1	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	O

2. 각 요구사항에 대한 구현방법

(1) Camera switcher

- FollowPlayer 스크립트, Main Camera와 연결됨

- left shift 를 누르면, 카메라가 전환됩니다.

- 카메라시점은 총 3가지로 다음과 같습니다.

- 하나의 Main Camera 오브젝트를 사용하여, 3가지 카메라 시점을 전환할 수 있습니다.

- 카메라 전환에 필요한 함수는 아래와 같습니다.
    1. CameraSwitch() - GetKetDown로 사용자가 왼쪽 시프트를 누르면, CameraNumber를 바꾸게 됩니다.

```
public void CameraSwitch()
{
    if (Input.GetKeyDown(KeyCode.LeftShift))
    {
        if (CameraNumber == 2) { CameraNumber = 0; }
        else { CameraNumber++; }
    }
}
```

- 이후, LateUpdate 함수에서 카메라 스위치 함수를 입력받아 CameraNumber를 통해 카메라의 시점을 전환합니다.

- 모든 카메라 시점은 플레이어의 자동차의 rotation이 변할 때, 카메라의 rotation도 변하게 하기 위하여, LookAt() 빌트인 함수를 사용하였습니다. 또한 카메라의 position값을 플레이어 자동차의 position값과 항상 일정한 거리만큼 떨어진 값으로 조정하기 위해 offset 좌표를 사용하였습니다.

- 아래는 카메라 시점 1의 코드입니다.

```
void LateUpdate() // LateUpdate -> 모든 스크립트의 UPdate가 실행된 이후 실행된다.
{
    CameraSwitch();

    if (CameraNumber == 0)
    {
        Vector3 offset;
        Vector3 lookAtPosition = Player.transform.position + new Vector3(0,2.5f,0);
        offset = -Player.transform.forward * 11f;
        offset.y += 10f;

        transform.position = Player.transform.position + offset ;
        transform.LookAt(lookAtPosition);
    }
```

- 나머지 카메라 시점들도 유사하게 동작합니다.

(2) Revival

- PlayerController 스크립트, Vehicle 과 연결됨
- 플레이어의 차가 도로 아래로 떨어졌을 때(정확히는 차의 y축 좌표가 -5.0f 이하로 줄어들때) 플레이어의 차를 인접한 도로 위로 다시 Revival 합니다.
- RevivalPoint1() 함수를 이용하여 플레이어의 차의 y축 좌표가 0 이하로 떨어지는 순간 차의position좌표와, 떨어지는 순간 차가 바라보는 방향의 방향벡터를 저장하고, Revival() 함수를 이용하여 플레이어의 차의 y축 좌표가 -5.0f 가 되는 순간, RevivalPoint1()에서 저장해둔 position 좌표와 방향벡터를 사용하여 다시 도로 위로 올려놓습니다.

- 아래는 두 함수의 코드입니다.
  RevivalRotation 변수를 사용하여, 떨어진 차가 도로 위로 올라갔을 때, 정확한 방향을 바라보게 했습니다.

- 떨어졌을 때, 다시 Revival 하는 부분은 목차 마지막의 영상 링크에서 확인할 수 있습니다

```
public void Revival_Point1()
{
    if (transform.position.y < 0 && transform.position.y> -0.05f )
    {
        RevivalPosition = transform.forward;
        Revivalpoint1 = transform.position + new Vector3(0, 0.3f, 0);
        RevivalRotation = transform.rotation;
        Debug.Log("Revival_Point1");
    }
}
참조 1개
public void Revival()
{
    if (transform.position.y < -5.1f)
    {
        RevivalRotation.x = 0f;
        RevivalRotation.z = 0f;
        transform.position = Revivalpoint1 - RevivalPosition*10;
        transform.rotation = RevivalRotation;

        Debug.Log("Revival_Point3");
    }

}
```

(3) Oncoming vehicles

- SpawnManager 스크립트, SpawnManager에 연결됨

- 플레이어의 차가 바라보는 방향의 반대 방향으로 총 16대의 차량이 다가오게 됩니다.
- SpawnCar() 함수를 사용하여, 활성화된 차량이 16대 가 될 때까지 차량을 spawn합니다. 여기서, 생성되는 차량은 Prefabs에 만들어져 있는 차량 Prefab을 사용합니다. SpawnCar 함수가 호출될 때마다 carIndex가 랜덤으로 초기화되어,(0부터 5) 무작위 차량을 생성합니다.

```
void SpawnCar()
{
    if(count < maxCarCount)
    {
        int carIndex = UnityEngine.Random.Range(0, carPrefabs.Length);
        Instantiate(carPrefabs[carIndex], spawnPoint, carPrefabs[carIndex].transform.rotation);
        count++;
    }
    return;
}
```

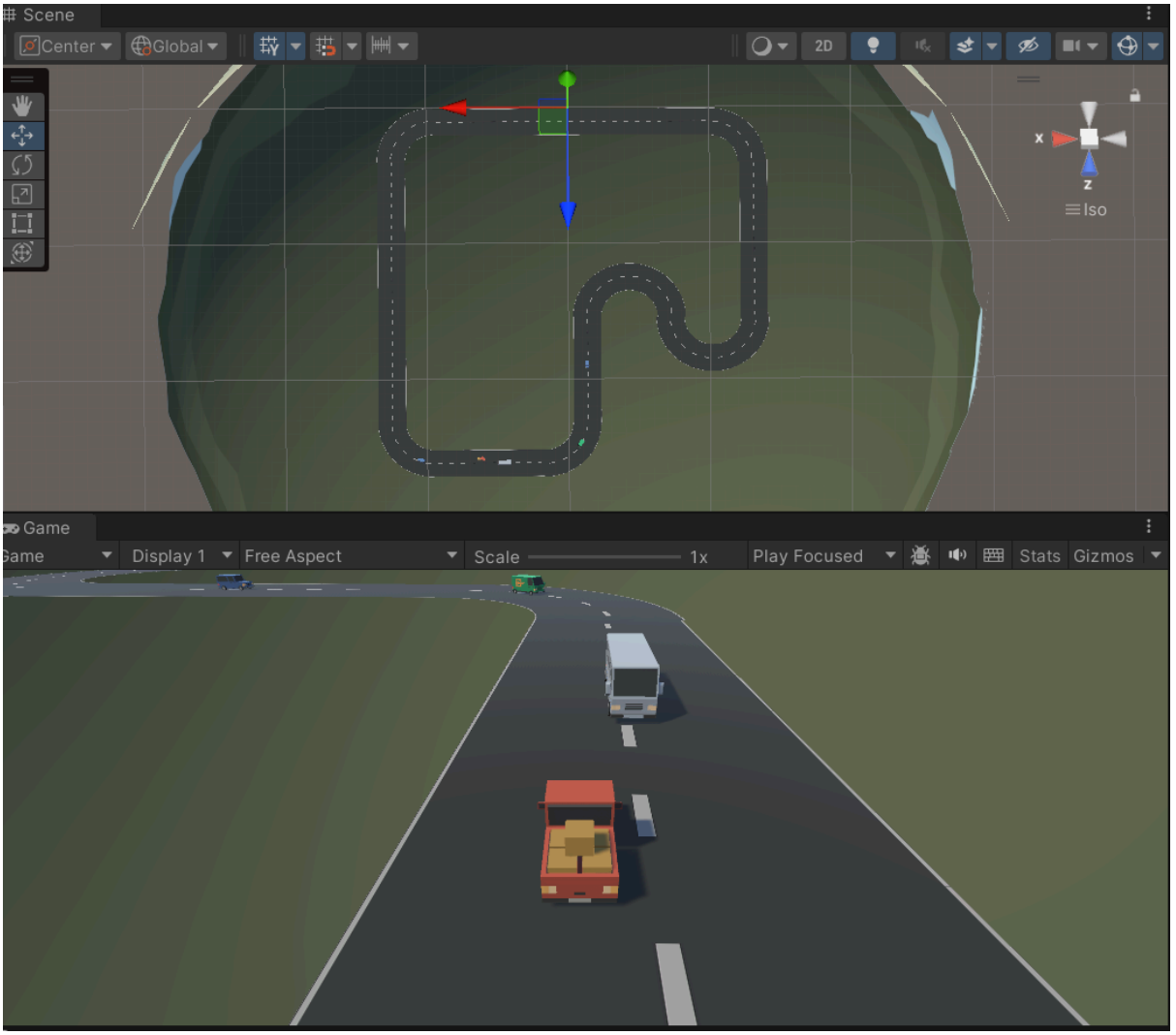
- 차량이 생성되는 위치의 좌표는 동일하고, 생성 주기를 6초(spawnInterval)를 두어, 6초마다 1대씩 총 16대의 차량이 도로 위에 생성됩니다.

```
void Start()
{
    spawnPoint = new Vector3(-13f, 0.01f, 144);
    nextSpawnTime = Time.time + spawnInterval;
}

// Update is called once per frame
Unity 메시지 | 참조 0개
void Update()
{
    if (Time.time >= nextSpawnTime)
    {
        SpawnCar();
        nextSpawnTime = Time.time + spawnInterval;
    }
}
```

-

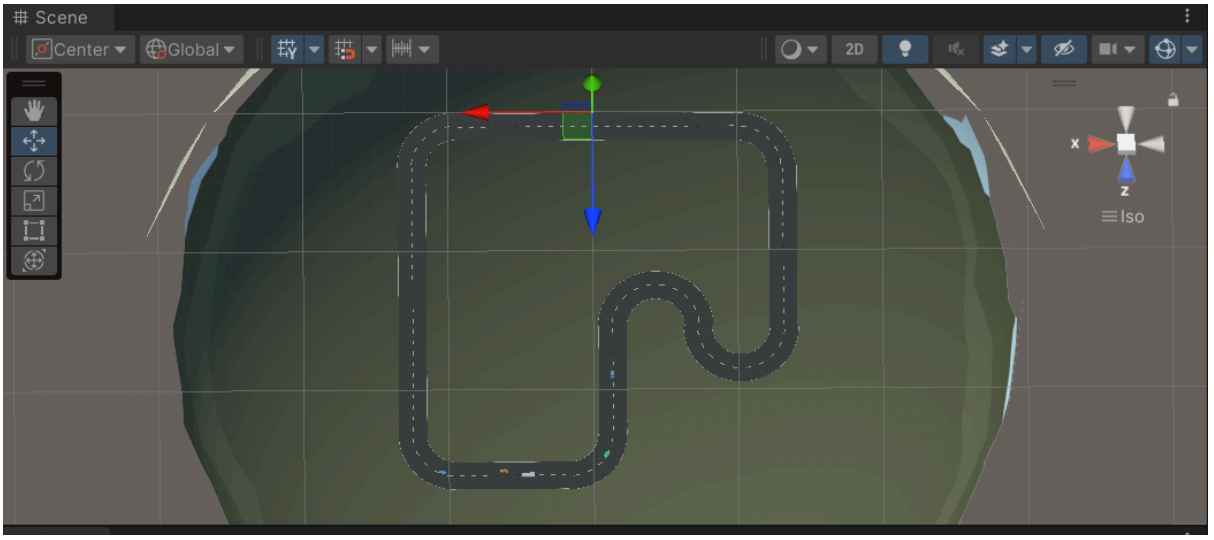
- 다음은 실행 결과입니다.


Scene
Center
Global
2D
Iso
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos

(4) Road layout

- MapGenerate 스크립트, MapGenerate와 연결

- 프리팹으로 저장해둔, 직선 가로 도로(wide), 직선 세로 도로, 커브 도로를 사용하여, 닫힌 구간의 도로를 생성합니다.
- 생성되는 도로의 레이아웃은 다음과 같습니다.



- H_Road(), V_Road, CurveRoad() 함수-인자(위치값, 생성할 도로의 갯수 또는 생성할 도로의 회전값)를 사용하여, 닫힌 구간의 도로를 생성합니다.(Index-> 도로 프리팹배열인덱스)

```
private void H_Road(int numberOfWideRoad, int VectorX, int VectorZ)
{
    for (int i = 0; i < numberOfWideRoad; i++)
    {
        Vector3 spawnPos = new Vector3(VectorX + (39 * i), 0.01f, VectorZ);
        int Index = 0;
        Instantiate(roadLayout[Index], spawnPos, roadLayout[Index].transform.rotation);
    }
}

참조 3개
private void V_Road(int numberOfWideRoad, int VectorX, int VectorZ)
{
    for (int i = 0; i < numberOfWideRoad; i++)
    {
        Vector3 spawnPos = new Vector3(VectorX , 0.01f, VectorZ + (39 * i));
        int Index = 1;
        Instantiate(roadLayout[Index], spawnPos, roadLayout[Index].transform.rotation);
    }
}

참조 8개
private void CurveRoad(float Rotate, int VectorX, int VectorZ)
{
    Quaternion rotation = Quaternion.Euler(0, Rotate, 0);
    Vector3 spawnPos = new Vector3(VectorX, 0.01f, VectorZ);
    int Index = 2;
    Instantiate(roadLayout[Index], spawnPos, rotation);
}
```

-

- start() 함수에, 세가지 함수를 적절히 호출하여 원하는 모양의 도로를 생성합니다. 아래는 그 예시입니다.

```
void Start()
{
    Resources.UnloadUnusedAssets();

    CurveRoad(90, 95, 40);
    CurveRoad(-90, 17, 232);
    H_Road(5, -100, 1);
    CurveRoad(0, 95, 232);
    V_Road(5, 115, 76);
    V_Road(2, -23,193);
    V_Road(3, -141, 78);
```

(5) Make a curved road mesh

- CurvedRoadMesh 스크립트, 프리팹의 CurvedRoad에 연결

- BaseMakeMesh를 상속받아, SetVertices(), SetNormals(), SetUV(), SetTriangles() 함수를 사용하여 곡선 도로를 만들었습니다.
- 곡선 도로를 만들기 위하여 두개의 사분원을 사용하였습니다. 두개의 사분원에 4.5도마다 Vertices를 등록하고, 이를 이용하여 삼각형 폴리곤을 만들고, 만들어진 삼각형 폴리곤을 조합하여 곡선 형태의 도로를 만들었습니다.
  마지막으로 만들어진 도로에, SetUV() 함수를 사용하여 Road 텍스쳐를 입혀 곡선 도로를 만들었습니다.
- setUV() 함수에서, 각 vertices의 시작점과 끝점에 대한 UV 좌표를 계산하여 리스트에 추가하고, 이를 통해 UV 매핑을 설정합니다

- 각 함수의 내용은 다음과 같습니다.

- 1. SetVertices() : degrees = 4.5f

```
protected override void SetVertices()
{
    radians = degrees * Math.PI / 180.0;
    for(int i = 0; i< 21; i++)
    {
        vertices.Add(new Vector3((float)Math.Cos(radians*i) * hsize, 0f, (float)Math.Sin(radians*i) * hsize));
        vertices.Add(new Vector3((float)Math.Cos(radians * i) * hsize*2, 0f, (float)Math.Sin(radians * i) * hsize*2));
    }
}
참조 2개
```

- 2. SetNormals()

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

- 3. SetUV()

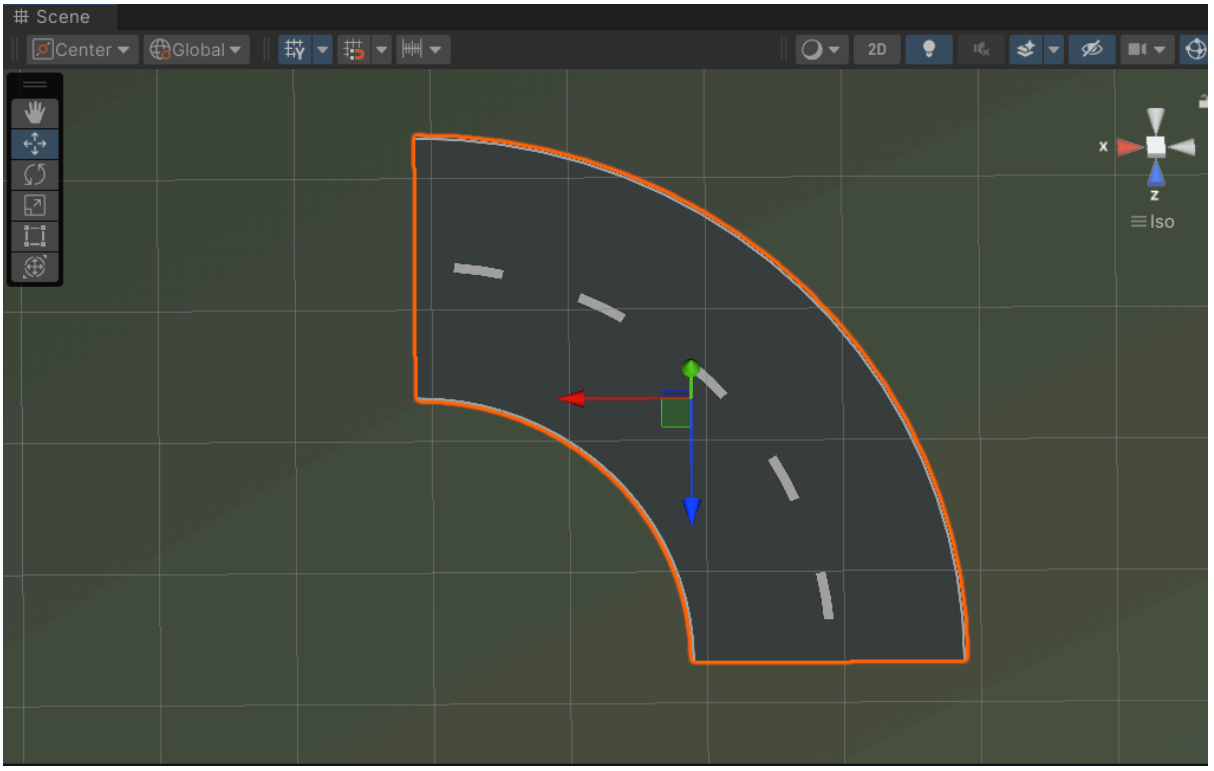
```
protected override void SetUV()
{

    float uvInterval = 1f / (vertices.Count / 2 - 1);
    for (int i = 0; i < vertices.Count / 2; i++)
    {
        float u = i * uvInterval;
        uv.Add(new Vector2(u, 0f));
        uv.Add(new Vector2(u, 1f));
    }
}
```

- 4. SetTriangles()

```
protected override void SetTriangles()
{
    for(int i = 0; i < 20; i++)
    {
        triangles.Add(2 * i);
        triangles.Add(2 * i + 3);
        triangles.Add(2 * i + 1);
    }

    for (int i = 0; i < 20; i++)
    {
        triangles.Add(2 * i);
        triangles.Add(2 * i + 2);
        triangles.Add(2 * i + 3);
    }
}
```

- 다음은 만들어진 곡선 도로의 모습입니다.
- 

(6) Oncoming vehicle on curved road

- MapGenerate 스크립트, Map Generate 에 연결
- AutoMoveFoward 스크립트, 차량 프리팹에 연결(플레이어 차는 아님)

- 플레이어의 차와 반대 방향에서 생성된 차량들이 도로를 따라 자연스럽게 움직일 수 있도록 하였습니다.

- 이를 위해, MapGenerate 스크립트와, SpawnManager를 통해 생성된 차량에 연결된 AutoMoveFoward 스크립트를 사용하였습니다.

- MapGenerate 에서는 도로의 정보를 담고 있는 roadDataList를 생성합니다. 이는 차량이 도로를 주행하기 위한 좌표들을 담고 있습니다. AutoMoveFoward 에서는 roadDataList를 기반으로 만들어진 wayPoints 리스트의 좌표값을 따라 차량들이 이동하게 됩니다.

- 이를 위해, RoadData 구조체를 사용하여,두 가지 유형의 RoadData를 생성하였습니다.
- 첫 번째로 직선도로Data, 두 번째로 곡선 도로Data입니다. 모든 도로Data에는 시작지점, 끝지점 좌표가 있습니다. 추가로 곡선 도로Data에는 곡선 위의 좌표가 18개 포함되어 있습니다.
- 다음은 RoadData 구조체의 두 생성자입니다.

```
public RoadData(Vector3 s, Vector3 e)
{
    degree = 0;
    radians = 4.5 * Math.PI / 180.0;
    start = s;
    end = e;
    curvePoints = null;
    center = Vector3.zero;
}

참조 8개
public RoadData( Vector3 o, int d)
{
    degree = d;
    radians = 4.5 * Math.PI / 180.0;
    start = Vector3.zero;
    end = Vector3.zero;
    curvePoints = new Vector3[19];
    center = o;
    CurvePoint(center, curvePoints, degree, radians);

}
```

- 다음은 곡선 위의 좌표를 생성하는 함수인 CurvePoint()의 일부입니다.

```
public void CurvePoint(Vector3 o, Vector3[] points, int d, double rad)
{
    if (d == 0)
    {
        start = o + new Vector3(-30f, 0f, 0f);
        end = o + new Vector3(0f, 0f, 30f);
        for(int i = 1; i < 20; i++)
        {
            points[i - 1] = o + new Vector3(-(float)Math.Cos(rad * i) * 30, 0f, (float)Math.Sin(rad * i) * 30);
        }
    }
```

- 이렇게 만들어진 도로Data들은 roadDataList에 저장되며, AutoMoveFoward 스크립트에서 roadDataList를 참조하여 waypoints 리스트를 생성합니다.

```
void Start()
{
    mapGenerator = GameObject.FindObjectOfType<MapGenerate>();
    waypoints = new List<Vector3>();

    for (int i = 0; i < mapGenerator.roadDataList.Count; i++)
    {
        RoadData roadData = mapGenerator.roadDataList[i];
        waypoints.Add(roadData.start);
        if (roadData.curvePoints != null)
        {
            foreach (Vector3 point in roadData.curvePoints)
            {
                waypoints.Add(point);
            }
        }
        waypoints.Add(roadData.end);
    }
```

- roadDataList의 원소에서 start와end를 리스트에 추가합니다. 만약, roadDataList의 원소가 곡선 좌표에 대한 정보를 가지고 있다면, 곡선 좌표들을 차례로 리스트에 추가합니다.

- 마지막으로, 차량이 waypoints리스트에 있는 좌표를 따라 움직이게 하는 함수입니다.(AutoMove())

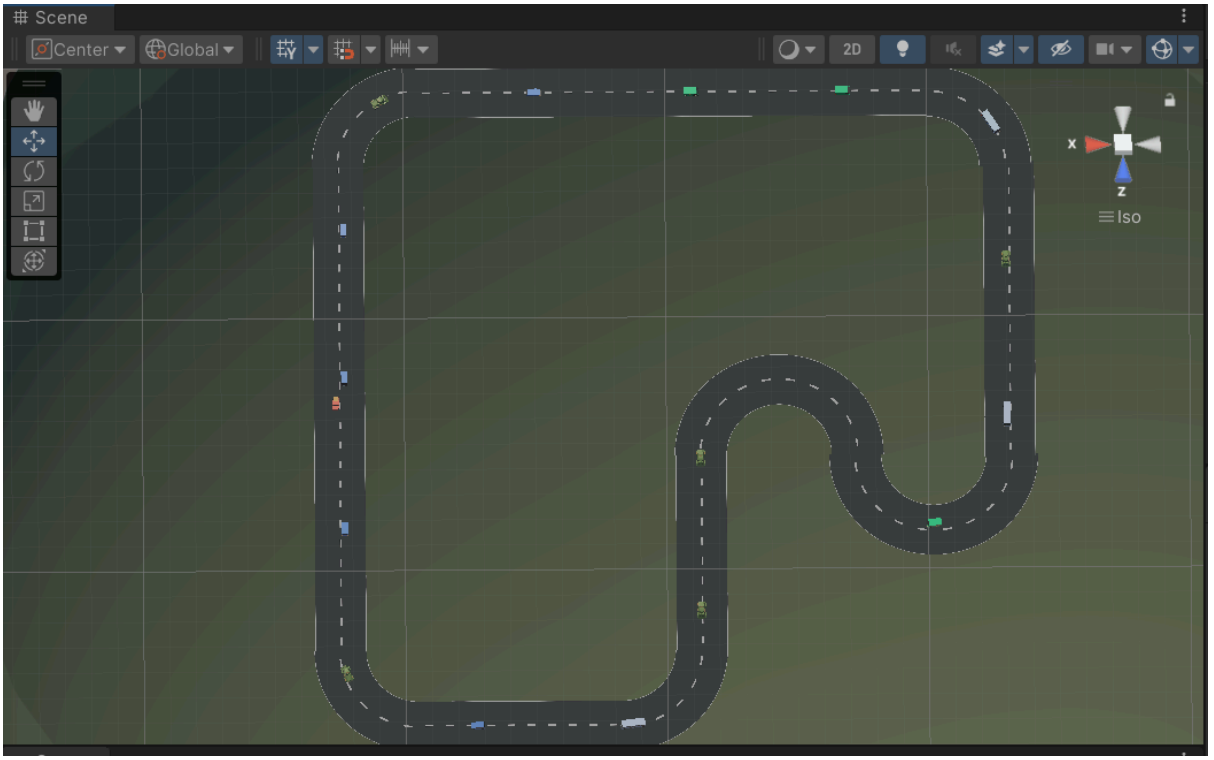
```
void AutoMove()
{
    if (waypoints != null && waypoints.Count > 1)
    {
        Vector3 targetPosition = waypoints[currentWaypointIndex];
        transform.position = Vector3.MoveTowards(transform.position, targetPosition, speed * Time.deltaTime);

        if (Vector3.Distance(transform.position, targetPosition) < 0.001f)
        {
            currentWaypointIndex++;
            if (currentWaypointIndex >= waypoints.Count)
            {
                currentWaypointIndex = 0;
            }
            transform.LookAt(waypoints[currentWaypointIndex]);
        }

    }
}
```

- 이 함수를 사용하여, 차량이 웨이포인트를 따라 움직이게 하고, 만약 모든 좌표를 지나쳤다면, 맨 처음 좌표부터 다시 따라가게 합니다.

- 도로를 따라 움직이는 차량의 사진입니다.



3. 과제에 대한 나의 의견

(1) 가장 구현하기 힘들었던 부분

- 곡선 도로를 만들 때, UV좌표를 이용한 텍스쳐 입히기가 의도한 대로 잘 동작하지 않아 조금 헤맸던 것 같습니다. 그래도 잘 동작하지 않던 원인을 찾아 해결해서 좋았습니다.

(2) 향후 이루고 싶은 목표

- 지금은 차량이 도로를 따라 움직일 때 주어진 좌표를 따라 움직여야 해서 좌표의 일부를 입력해야 했지만, 향후에는 자동으로 좌표를 생성할 수 있었으면 좋겠습니다.

4. 완성본 동영상 링크

https://youtu.be/bgf_gQy3Y74