가상세계
Driving Car
과제 보고서

2024.4.16

2020203052
소프트웨어학부 권빈

Easy 1	Camera Switcher	
Easy 2	Revival	
Medium 1	Oncoming Vehicles	
Hard	Make a Curved Road Mesh	
Medium 2	Road Layout	
Expert	Oncoming Vehicle on Curved Road	

1. Camera Switcher

```
void LateUpdate()
{
    if (Input.GetKeyDown(KeyCode.LeftShift) || Input.GetKeyDown(KeyCode.RightShift))
    {
        changecam++;//카메라 시점을 설정하는 int형 변수
    }
    if((changecam%4)==0){ //첫번째 시점
        transform.position=player.transform.position+new Vector3(0,5,-8);
        transform.rotation = Quaternion.Euler(10, 0, 0);
    }else if((changecam%4)==1){ //두번째 시점
        transform.position=player.transform.position+new Vector3(-10,5,0);
        transform.rotation = Quaternion.Euler(10, 90, 0);
    }else if((changecam%4)==2){ //세번째 시점
        transform.position=player.transform.position+new Vector3(7.5f,5,0);
        transform.rotation = Quaternion.Euler(10, -90, 0);
    }else{//네번째 시점
        transform.position=player.transform.position+new Vector3(0,5,10);
        transform.rotation = Quaternion.Euler(10, 180, 0);
    }
}
```

chagecam 이라는 변수를 왼쪽 혹은 오른쪽 shift를 누를때마다 1씩 증가시켜
chagecam % 시점 개수의 값에 따라 정해진 시점으로 변경한다.

2.Revival

```
    timer+=Time.deltaTime;//현실시간으로 timer값 증가

    if(timer>4){//timer가 4이상일 때 ResetPosition함수 실행 후 타이머 초기화
        ResetPosition();
        timer=0;
    }

    if(transform.position.y<-2){// 객체의 y위치가 -2보다 떨어졌을 때 저장했던 위치로 이동
        transform.position = currentPosition;
        transform.rotation = currentRotation;
    }

}

참조 1개
void ResetPosition()//현재 오브젝트의 위치를 저장
{
    currentPosition = transform.position;
    currentRotation = transform.rotation;
}
```

timer 변수에 Time.deltaTime을 더하면서 실제 시간이 4보다 넘었을때 ResetPosition을 실행시킨다.
ResetPosition은 자동차의 현재 위치를 저장해두고
매 4초정도 마다 위치를 갱신하면서 자동차의 position의 y가 -2보다 낮아지면
즉 떨어졌을 때 저장했던 위치로 돌아간다.

3. Oncoming Vehicles

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

단순히 자동차의 반대방향에서 오면 되기 때문에 오브젝트를 기준으로 정해진 speed로 앞으로 가는 코드를 작성한뒤 player 자동차의 반대방향으로 rotation을 설정하는 방식으로 문제를 풀었다. 그 뒤 자동차 오브젝트를 일정 거리를 두고
3개를 배치했다.

```
    if(transform.position.y<-10){
        Destroy(gameObject);
    }
}
```

자동차가 떨어지면서 계속 위치를 새로 갱신하면서 메모리를 잡아먹는것을 방지하려고 일정이상 떨어지면 오브젝트를 제거하는 스크립트를 썼다.

4.Make a Curved Road Mesh

```
참조 2개
protected override void SetVertices(){
    float angleStep = 90.0f / dense;
    for (int i = 0; i <= dense; i++)
    {
        float angleInRad = Mathf.Deg2Rad * (angleStep * i);
        float x = Mathf.Cos(angleInRad) * size;
        float z = Mathf.Sin(angleInRad) * size;
        vertices.Add(new Vector3(x, 0f, z));
        x = Mathf.Cos(angleInRad) * hsize;
        z = Mathf.Sin(angleInRad) * hsize;
        vertices.Add(new Vector3(x, 0f, z));

    }
}
```

90도를 dense 나누어 dense를 곡선의 곡률을 정하는 변수로 만들고 해당 각도를 Mathf.Deg2Rad을 통하여 라디안 각도로 바꾸어준다. 이후 곡선의 바깥쪽 정점은 위에서 라디안각도로 만들어준 angleInRad를 이용하여 x좌표는 size*cos(angleInRad) z좌표는 sin*(sin(angleInRad)로 설정해준뒤 정점으로 만들어주고 안쪽 정점은 size를 hsize로 바꾸어 폭이 hsize인 곡선을 만들 정점을 설정해준다.

```
참조 2개
protected override void SetNormals(){
    for (int i = 0; i < vertices.Count; i++)
    {
        normals.Add(new Vector3(0f, 1f, 0f));
    }
}
```

SetNormals함수는 위에서 설정한 정점의 개수만큼 (0,1,0)
즉 평면에수직으로 설정해주었다.

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

각 정점이 존재하는 위치에 맞게 최대1 비율로 맞춰 uv를 설정해주었다.
따라서 개수 역시 정점의 개수와 같게 범위를 맞춰주었다.

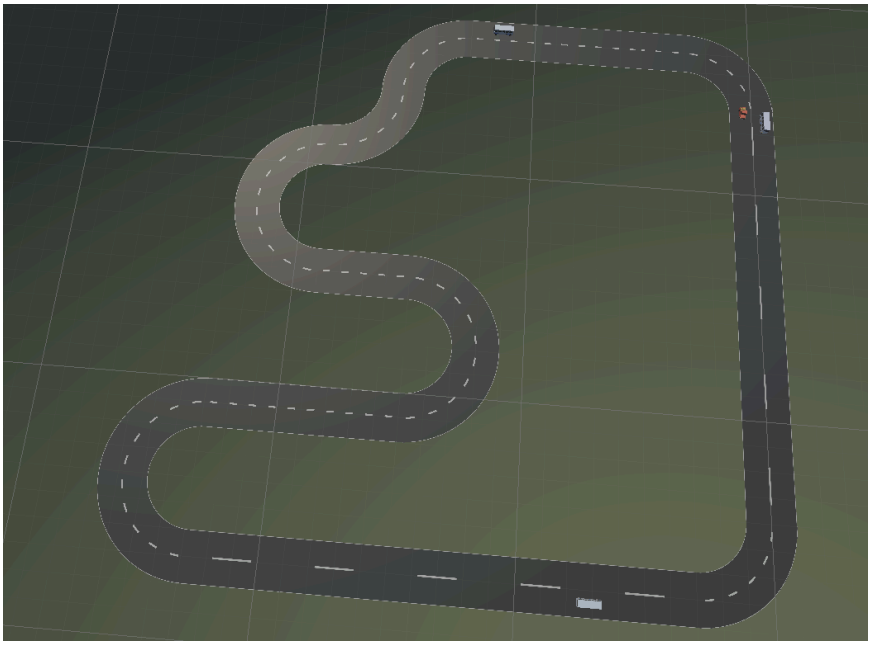
```
참조 2개
protected override void SetTriangles(){
    for (int i = 0; i < dense * 2; i += 2)
{

    triangles.Add(i);
    triangles.Add(i + 1);
    triangles.Add(i + 2);

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

위 코드에서 첫번째 삼각형을 과제자료의 0,2,3 삼각형모양
두번째 삼각형을 1,2,3삼각형모양으로 설정하여
정점이 4개가되어 사각형을 이룰때마다 삼각형 두개를 생성하여 연결할 수 있도록 코드를 구성하였다. 이때 중복되는 경우를 방지하기 위해 dense의 범위를 두배로 늘리고 i의 증가량을 2로 만들었다.

5. Road Layout



위에서만든 curved road와 straight road 또 전에 받았던 road를 이어붙여 Road LayOut을 만들었다

6.Oncoming Vehicle on Curved Road

```
public void StartRotation()
{
    // 현재 회전에서 90도 추가한 것을 목표 회전으로 설정
    targetRotation = Quaternion.Euler(transform.eulerAngles.x, transform.eulerAngles.y + turn, transform.eulerAngles.z);
    startRotation = true; // 회전 시작
}
```

먼저 StartRotaion함수다. targetroation을 설정하고 rotation변경을 시작하는 startRotation 변수값을 true로 만들어준다.

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


    if (startRotation)
    {
        // Quaternion.Slerp를 이용해 점진적으로 회전
        transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, Time.deltaTime * rotationSpeed);

        // 현재 회전이 목표 회전에 충분히 가까운지 확인
        if (Quaternion.Angle(transform.rotation, targetRotation) < 0.01f)
        {
            // 회전 완료
            transform.rotation = targetRotation; // 정확한 목표 회전 각도로 설정
            startRotation = false; // 회전 중지
        }
    }
```

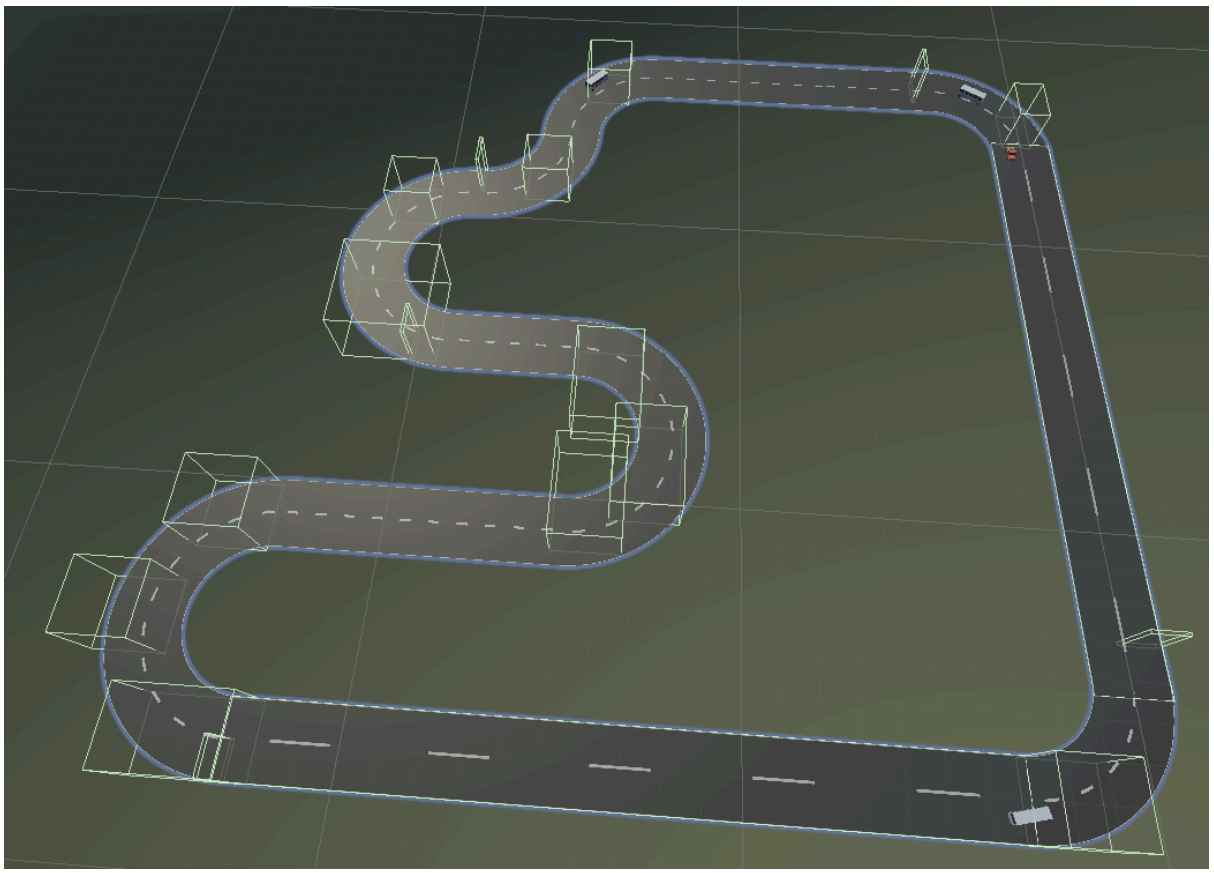
특정조건으로인해 StartRoataion 함수가 작동하면 startRoation값이 true로 바뀌고 Update안의 if문이 작동한다. 이때 목표 방향만큼 부드럽게 회전하게하기위해, Quaternion.Slerp과 변화율을 현실시간으로 바뀌게 하였다.
회전이 끝나면 startRotaion을 false로 바꿔 회전을 중지한다.

```
private void OnTriggerEnter(Collider other) {
    if(other.tag=="turnleft"){
        turn=-45;
        StartRotation();
    }else if(other.tag=="turnright"){
        turn=45;
        StartRotation();
    }else if(other.tag=="set90"){
        transform.rotation = Quaternion.Euler(0, 90, 0);
        startRotation=false;
    }else if(other.tag=="set0"){
        transform.rotation=Quaternion.Euler(0,0,0);
        startRotation=false;
    }else if(other.tag=="set-90"){
        transform.rotation=Quaternion.Euler(0,-90,0);
        startRotation=false;
    }else if(other.tag=="set180"){
        transform.rotation=Quaternion.Euler(0,180,0);
        transform.position = new Vector3(-105, transform.position.y, transform.position.z);
        startRotation=false;
    }else if(other.tag=="set-90x"){
        transform.rotation=Quaternion.Euler(0,-90,0);
        transform.position = new Vector3(98, transform.position.y, 149.8f);
        startRotation=false;
    }
}
참조 2개
```

투명 큐브를 road에 설치한뒤 콜라이더로만들어 자동으로 움직이는 자동차들이 큐브를 만났을때 큐브에 종류, 즉 태그에따라서 다르게 행동하는 방식으로 만들었다

turnleft와 turnright는 각각 왼쪽으로 45도 오른쪽으로 45도 설정한뒤 StartRotaion함수를 실행시키는데 90도가아닌 이유는 한번에 너무 회전시키는것보다 나눠서 회전시키는것이 더 적합해서 45도로나눈뒤 큐브 오브젝트를 나누어 배치하였다.

그아래에 나열된 set(각도) tag는 큐브오브젝트와 만났을시 자동차의 rotation값을 고정적으로바꾸어주는데 위의 turnleft와turnright로 회전시 정확히 각도가 0,90,180,270으로 설정되지않아 주행거리가 늘어나면 도로를 벗어나 떨어지는 문제가 생김을 방지하기위해 일정 커브마다 배치해두었다.
이중에는 좌표까지 설정해두는것이 있는데 이또한 자동으로 움직이는 자동차가 계속해서 도로를 달리게하기위해 일정 발판마다 자리를 잡아주는 용도로 사용하였다.



실제 길에 투명큐브를 배치한 모습

과제 자기평가

과제를 하면서 가장 많이느낀점은 배운것만 가지고 하기 힘들다는 점이였다.
그래서 이것저것 찾아보고 실험해가면서 과제를 수행하며 생각했던것보다
실력이 많이 는것 같다. 또 대체로 만족스럽게 구현했다는 점이 잘한점이지만
마지막 과제인 Oncoming Vehicle on Curved Road를 구현하는 과정에서 큐브를 로드에 하나하나 배치하는 방법이아닌 더 깔끔한 방법으로 구현하고싶었지만 실패하였고 그나마 구현한 방법도 자율주행버스의 움직임이 완전히 자연스럽지 않은점이 보완하고 싶은 점이다.

시연영상링크

https://youtu.be/LoFbsjmOSD8