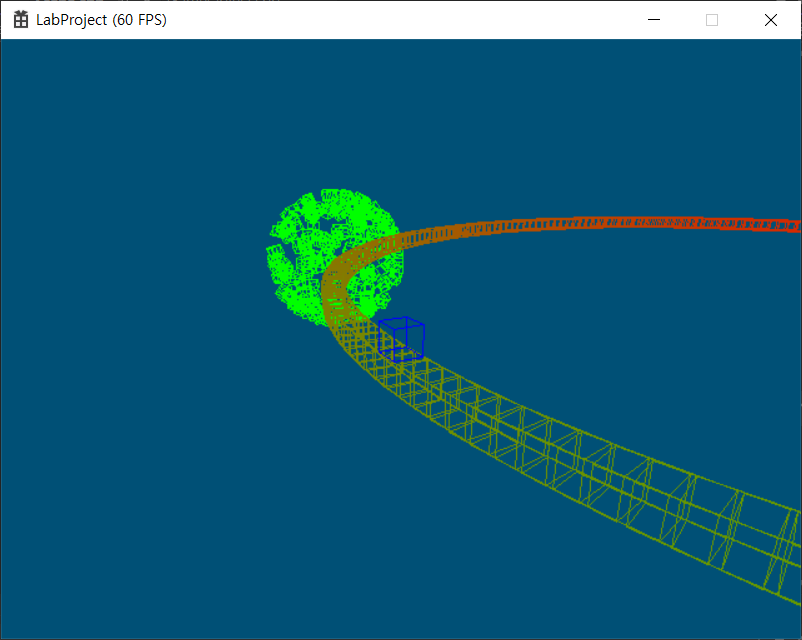
3D 게임프로그래밍

과목명: 3D 게임프로그래밍  
학번: 2020182048  
이름: 김정학

실행화면



레일을 깔기 시작했을 때, 직선으로만 레일을 몇 개 깔고 플레이어가 W버튼을 누르면 플레이어의 위치를 다음 레일로 옮기는 식으로 플레이어를 움직였고, 이때 레일들을 Scene클래스의 object들의 배열에 집어넣고 W를 누를 때마다 index값을 늘려 해당 index에 맞는 레일의 좌표를 가져와서 플레이어 좌표를 바꾸었다. Index를 레일의 개수로 나눈 나머지를 배열의 번호로 사용하면 처음부터 마지막까지 순환하게끔 만들 수 있었다. 하지만, 롤러코스터라면 레일의 시작부분과 마지막부분이 이어져 있어야 한다고 생각해서 레일을 object의 멤버함수 rotate로 회전시켰더니, 무조건 자전만 되게 행렬 곱이 되어서 공전하게끔 행렬을 Translate, Rotate 순으로 곱해서 world matrix를 지정해주었다.  
 이러한 방법으로 설치한 레일들이 20개, 30개 많아지면 Scene의 코드가 굉장히 길어진다. 그래서 Track이라는 클래스를 만들고 멤버로 게임 오브젝트를 담은 vector를 만들고, 만든 모든 레일들을 그 vector에 담고 인덱스를 통해 접근하는 방법을 택했다. 출력은 잘 되었지만 처음에는 Render 함수의 Frustum체크에 걸려서 모든 오브젝트들이 다 같이 사라지는 마법 같은 일이 일어났다. 체크하는 함수를 따라가보니 오브젝트들의 OOBB를 가지고 체크를 하는 것을 확인하였고, 이 OOBB가 업데이트 되지 않아서 플레이어가 특정 지점을 지나가면 모든 레일들이 한 번에 그려지지 않는 버그가 발생한 것이었다. 따라서, UpdateBoundingBox라는 오브젝트 클래스의 멤버 함수를 사용해서 각 객체마다의 OOBB를 업데이트 하였다. 또한, 레일을 동일한 색으로 생성하니 움직일 때 구분이 잘 되지 않아서 레일마다 조금씩 색을 바꿔주었다.  
 조작키는 R과 W, S에만 특정 동작들을 수행하게 하였고, 각각 R은 깔려 있는 레일 중 첫번째 레일. 즉, 시작점으로 이동하는 키이다. W키를 누르면 플레이어가 움직이고 있음을 나타내는 isCoasterMoving이라는 새로 만든 bool 변수의 값을 true로, S는 false로 만들어 주는 행동을 하게 하였다. 만약 플레이어가 움직이고 있을 때, R키를 눌러서 위치만을 바꾸어 주면 플레이어가 어떤 레일 위를 지나가고 있는지 나타내는 index값이 초기화 되지 않기 때문에, 이상한 방향으로 돌아가게 되므로 R키가 눌렸을 때 index의 값을 0으로 초기화 시켜주는 작업도 같이하였다.  
 롤러코스터가 움직이는 부분은 Scene 클래스 안에서 처리하였는데, Scene 안에서 매 루프마다 불리는 함수는 Render와 Animate로 보았고, 롤러코스터의 속도는 매 루프마다 계산해주어야 하기 때문에 둘 중 좀 더 맥락상 맞는 것 같은 Animate에서 하였다. Animate 함수 내에서 조건으로 앞서 말한 isCoasterMoving이라는 bool 변수를 통해 만약 롤러코스터가 움직이고 있다면, 몇 번째 레일 위를 지나고 있는지 나타내는 index값을 통해 앞의 레일과 현재 있는 레일의 위치를 통해 방향을 구하고, 롤러코스터에 해당하는 플레이어의 멤버 함수 Move를 통해 방향으로 움직이게 하였다. 이렇게만 하면 플레이어가 처음보는 방향 그대로 레일을 따라 움직이기 때문에 멤버 함수 중 하나인 LookTo를 사용해서 움직이는 방향과 동일한 방향을 바라보게 하였다.  
 이후에는 적 오브젝트와 충돌하면 적을 파괴하는 기능을 구현하고 싶어, 교수님께서 만들어 두신 ExplosiveObject를 생성하여 트랙의 중간에 위치시켰다. 다음으로는, 플레이어와 적 오브젝트의 충돌하는 함수를 구현하였는데, 이 함수 내에서는 플레이어와 오브젝트 각각의 OOBB를 Intersect함수를 통해 충돌 여부를 검사하고, 충돌이 일어난다면 적 오브젝트를 터트리게 하였다. 결과적으로는 잘 작동하였고, 충돌 조건 또한 매 루프마다 잘 리셋 되었다. 오브젝트 간의 충돌 검사와는 다르게 플레이어는 하나뿐이므로 이중 루프도, 루프를 두 번 하는 작업도 할 필요가 없다고 생각되어서, 모든 오브젝트들을 한번 루프하고 그 중에 충돌하는 오브젝트가 존재한다면 바로 해당 오브젝트가 폭발하게 하였다. 그리고, 모든 충돌 체크 함수들이 Scene 클래스 내의 m\_ppObjects라는 오브젝트들의 배열에 들어있는 원소들을 가지고 체크를 하기 때문에, Track 클래스에 따로 만들어 둔 레일들은 오브젝트 충돌 대상이 되지 않기 때문에 걱정할 필요가 없다고 생각했다.  
 과제를 하면서 플레이어의 뒤에 카메라를 따라가게 하면서 플레이어의 이동 방향을 바라보게 하고 싶어서 카메라의 LookAt 함수를 사용해 보았는데, 의도하지 않은 결과가 나왔고 해결하기 위해서 구조를 살펴 보았지만 원하는 대로 작동하게끔 고치진 못하였다.