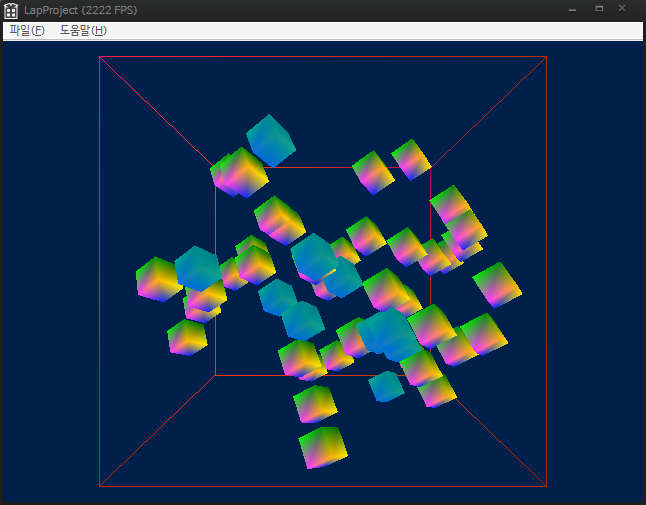
게임공학과 2012180004 권창현



1) 과제 완성 사진 및 과제를 완료하지 못한 부분

과제를 최종적으로 완료하면서 완료 하지 못한 부분은 마우스를 클릭하였을 경우 레이저가 발사되어 마우스 클릭한 위치에서 가장 가까운 Z축 오브젝트를 지운다.

2) 프로그래밍 과정

우선 처음으로 시작을 한 것은 그동안 해온 따라하기를 진행하는 것 이였다. 올라온 따라하기를 보다가 따라하기 16까지를 진행을 하면 요번 구현 내용들은 모두 구현을 할 수 있을 거 같다는 생각을 하고선 작업을 진행하였다.

2-1) 오브젝트 개수 줄이기 & 생성하기

따라하기 16까지를 완료한 후 가장 처음으로 해당 오브젝트들의 위치를 조절 하는 것이었다. 따라하기 9까지 의 경우 모든 오브젝트 애니메이션 이나 추가 부분을 Scene.cpp 에서 진행을 하였지만 따라하기 16에서는 오브젝트의 생성은 Shader의 BuildObjects 에서 진행을 하고, 애니메이션 부분은 Object에서 진행을 한다. 처음 진행 시 오브젝트를 한 개만 만들어서 정상적으로 내가 보는 위치에 나타내게 하기 위하여 void CInstancingShader::BuildObjects( ID3D11Device \*pd3dDevice ) 함수의 m\_nObjects를 1로 수정을 하였다. 하지만 1개의 객체가 나오지 않고, 아예 오브젝트가 생성하지 않는다. 이후 숫자를 여러 번 늘려서 진행 해본 결과 17이 되어야지 1개의 오브젝트가 정상적으로 출력이 되었다.

왜 오브젝트를 17개를 만들어야지 1개가 작성이 되는지 확인을 하다가 Render 부분에서 cube오브젝트를 만들기 전에 Sphere 오브젝트를 만들기 위해서 미리 16개의 오브젝트를 만드는 부분에서 이러한 문제가 발생하였다. 따라하기 16에서 사용하는 부분이라서 과감하게 삭제를 한 뒤 내가 필요한 m\_nObjects 개수만 받아와서 오브젝트를 렌더링 할 수 있게 수정을 하였다.

2-2) 오브젝트 움직이게 하기 위한 작업

정상적으로 오브젝트가 출력되는 것을 확인 후, 해당 오브젝트의 현재 위치인 SetPosition 과 어디로 움직여야 할지를 제공하는 setDirection을 클래스에 추가를 해주었다. SetDirection 같은 경우 추 후 법선 벡터를 이용한 충돌 체크를 진행하기 위하여 Vector 로 값을 받아 왔다. 이후 해당 객체를 랜덤 하게 움직이게 하기 위하여 STL에서 배운 default\_random\_engine를 사용하여 움직이는 방향 과 +, - 위치를 setDirection에 집어 넣어 주었다.

해당 오브젝트를 움직이게 하기 위하여 Object.cpp 에서 Animate 부분에 들어가 m\_d3dmtxWorld 와 moveDirection 을 더하여 오브젝트 위치를 이동시키게 해주었다.

2-3) 큰 큐브 오브젝트 만들기

그 뒤 큰 큐브를 제작하기 위하여 지금까지 만든 오브젝트 클래스인 Mesh.h 에 들어가서 CCubeMesh 를 복사하여 CBCubeMesh 를 만들어 큰 큐브를 만들 수 있게 작업을 하였다. 작업하면서 기존의 작은 큐브 들은 D3D11\_PRIMITIVE\_TOPOLOGY\_TRIANGLELIST 를 사용하여 육면체를 삼각형 12개로 만들었으나 큰 큐브를 그렇게 만들 경우 큐브 모습이 정상적인 큐브 모습이기는 하나 삼각형 12개가 모두 보이게 되어 D3D11\_PRIMITIVE\_TOPOLOGY\_LINESTRIP 를 이용하여 사각형으로 큰 큐브를 만들게 되었다. 라인스트립을 이용하면서 기존 삼각형 12개로 만들어진 m\_pnindices를 수정을 해주어야 했다. 삼각형의 경우 18개의 위치가 들어가지만, 사각형은 이보다 많은 20개의 위치를 잡아주어 정확한 육면체를 만들 수 있게 되었다. 이후 Fill 모드가 Solid 형태가 아닌 WireFrame 모드로 변경을 하여 속이 제대로 보이는 큰 큐브를 만들 수 있는 클래스가 완성이 되었다.

그 이후에는 큐브 오브젝트를 그려주는 거와 동일하게 Shader.cpp 에서 BuildObjects 와 Render를 추가해주는데 일반 작은 큐브와는 다르게 크기를 15.0f 크기로 오브젝트를 생성하였다.

2-4) 큐브 오브젝트의 충돌 체크

이제 해당 큰 큐브에 작은 큐브가 충돌할 경우 튕겨져 나가는 부분을 해결하기 위하여 법선 벡터와, 반사 벡터를 이용하여 작업을 하였다.

충돌 체크 같은 경우는 Shader.cpp 에서 코드를 작성하였다. 기존 1차 과제와 똑 같은 내용이라 자세한 설명을 제외하고, 수정된 내용이 있다면, 우선 큐브 크기가 15.0f 에 0.5가 추가 되어 총 17.5f 크기의 큐브로 충돌 체크 범위를 변경 하였다. 이후 작은 큐브의 현재 위치인 m\_d3dmtxWorld 매트릭스로 되어 있어, 충돌 체크를 위하여 Vector 변수를 임의로 선언을 하여 충돌 체크를 한 뒤 충돌이 되었을 경우 반사 벡터를 구하여 moveDirection에 해당 값을 전달하여 제대로 된 반사 벡터를 구현 하였다. 이 와 같은 충돌 체크를 총 6면을 구현하여 모든 면에서의 충돌 체크를 완료 하였다.

2-5) 큐브 마우스 피킹 작업

마우스 피킹을 위하여 현재 클릭 된 마우스 위치 좌표를 뷰 포트 역 변환, 투영 역 변환, 뷰 역 변환 하여 검사를 하면 정상적으로 피킹을 할 수 가있었다. 따라하기 16까지 진행을 하게 되면 이 부분에 대해서 모두 작업이 완료가 된다. 이 후 z축 검사를 하여 가장 가까이에 있는 큐브를 먼저 지울 수 있게 작업을 할 수가 있다.

피킹이 되어 해당 오브젝트가 확인이 되면 IsVisible bool 값이 변하게 된다. 이것을 Shader.cpp 에서 Render부분이 해당 오브젝트를 그릴지 말지를 판단하여 실제로 클릭이 되면 사라지는 것처럼 보이게 되는 것이다. 화면상에서는 사라진 것처럼 보이지만 실제로는 해당 오브젝트는 계속 충돌 하면서 움직이고 있다.

3) 프로그래밍 해결 못한 부분

마우스 피킹시 해당 위치로 레이저가 날라가게 하는 부분은 구상한 바로는 클릭 시 현재 클릭 된 좌표 위치와 z 축 값을 알고 있기 때문에 해당 위치로 레이저를 그리면 될 거라 생각 하였다. 하지만 막상 작업을 하였더니, 해당 위치로 레이저가 날라가지 않고 다른 위치로 날라가거나 아예 나오지 않았던 부분도 있었다. 아무래도 z축 값이 정확하게 나오긴 하지만 오브젝트를 그려주면서 어디선가 그려지는 부분이 잘못 건드려 지는 거 같다.

4) 조작 방법

마우스 피킹을 사용하여 레이저 포인트가 나가지 않지만, 클릭을 하면 해당 오브젝트가 정상적으로 지워진다.

5) 과제를 마치면서

교수님이 말씀하신대로 2주동안 따라하기 와 어떻게 프로그램을 작성 해야할지 고민을 하고 1주동안 코딩을 하면 프로그램을 완성할 꺼라 생각 하였지만 생각보다 시간이 좀 더 오래 걸렸다. 지각 제출은 어쩔 수 없지만, 그래도 너무 늦게 내는 것 보다는 빠르게 내는 것이 옳 은거 같아서 최대한 내가 할 수 있는 부분까지 완성을 하여 과제를 제출 하게 되었다. 곧 과제3도 완성을 해야하는데 이번에는 지각 제출하지 않고선 구현된 내용을 모두 완성할 수 있게 작성하여 제출을 해야겠다.