Basic OpenGL Viewer

컴퓨터그래픽스 - 컴퓨터소프트웨어학부 2017029343 김기환

Which requirements you implemented

1. Orbit.

- azimuth는 마우스의 움직임에 따라 변하는 x값에 따라 g_cam_ang 값을 변화시켜 이를 Camera Position에 곱하는 식으로 구현하였습니다.
- elevation은 마우스의 움직임에 따라 변하는 y값에 따라 g_cam_height 값을 변화시켜 이를 카메라에서 원점으로 향하는 방향벡터의 수직인 축을 기준으로 glm.rotate()를 통해 구현하였습니다. y축의 꼭대기에 도달했을 경우 다시 내려오는 식으로 구현하였습니다.

2. Pan.

- target_cam_trans_x와 target_cam_trans_y의 값을 마우스 움직임에 따라 변화하는 x값 y값을 통해 얻어낸 후, 이를 glm.translate() 라이브러리 함수를 통해서 이동변환행렬을 구한 후, 이를 Camera Frame에 곱하는 식으로 구현하였습니다.

3. Zoom.

- 마우스 휠에 따라 변화하는 g_cam_zoom 변수의 값을 통해서, glm.translate()함수의 인자로 전달하여, 이를 통해 얻은 이동변환행렬을 Camera Frame에 곱하는 식으로 구현하였습니다. 이는 곧 Camera가 w 축을 따라 앞 뒤로 이동함을 의미하게 됩니다.

4. Mouse Movement and KeyBoard Input.

- 마우스 드래그, 마우스 휠, 키보드 입력에 대한 콜백함수를 정의하여, 함수 내부에서 각 기능에서 사용될 변수들의 값을 변화시켜 주도록 하였습니다.
- 마우스와 관련된 콜백함수에서는 윈도우에서의 이전 마우스의 위치와 현재 마우스의 위치의 변화를 통해 g_cam_ang, g_cam_height, target_cam_trans_x, target_cam_trans_y의 값을 변화시켜주었으며, 마우스 휠도 동일한 방식으로, g_cam_zoom 값을 변화시켜주었습니다. 키보드에서 v키를 입력하게되면 전역변수로 선언한 Bool 변수인 isPerspective의 값을 토글시켜주어 Perspective ↔ Othogonal을 구현하였습니다.

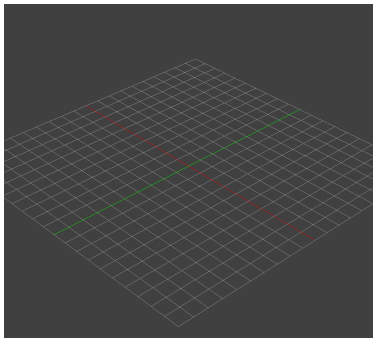
5. Perspective / Orthogonal projection.

- glm.ortho(), glm.perspective() 라이브러리 함수를 통하여 MVP Matrix의 P 부분을 계산하여 구현하였습니다.
- Perspective Projection에서 줌 기능을 통해 카메라를 이동 시키면 앞 뒤로 물체가 커졌다가 작아졌다 합니다. 이를 통해 원근감을 느낄 수 있었지만, Orthogonal Projection에서는 카메라의 위치와 관계없이 일정한 크기를 유지하게 되어 줌 효과를 느낄 수 없었습니다. 줌 기능을 통해 확인한 결과 View Volume 밖으로 물체가 이동하게 되면 윈도우를 통해 사용자는 Object를 관찰할 수 없음을 알게 되었습니다.

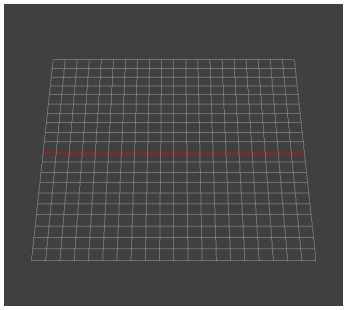
6. Rectangular Grid with lines.

- DrawPlaneXZ()라는 렌더링함수를 만들고, 해당 함수에서 VAO, VBO를 설정하여 VAO를 리턴하고 메인 함수의 반복문에서 윈도우에 그리도록 하였습니다.
- DrawPlaneXZ()에서 반복문을 통해, x축, z축을 그리되 각각 총 20개의 라인을 그리도록 하였으며, 블렌더와 동일하게 보일 수 있도록 원점을 지나는 라인은 색상을 변경해주었습니다.

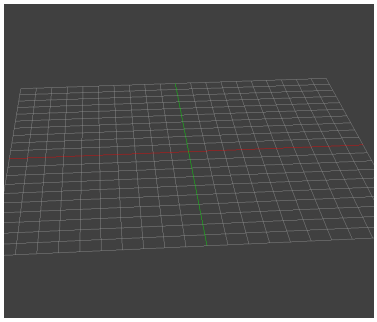
A few screenshot images of your program



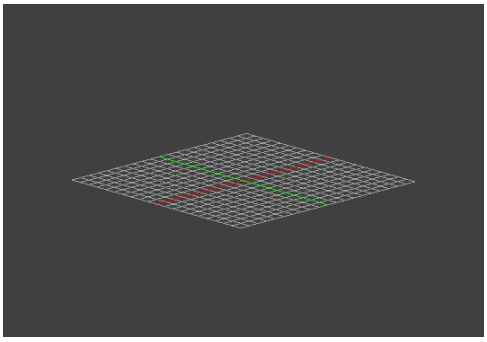
Perspective #1



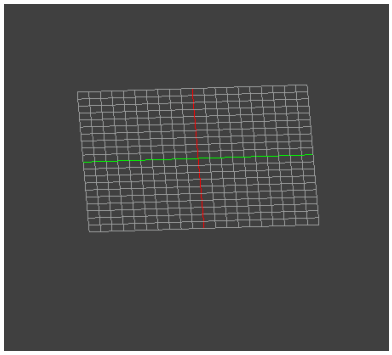
Perspective #2



Perspective #3



Orthogonal #1



Orthogonal #2