Trackball_OpenGL

윗몸일으키기 모델을 바라보는 camera 를 조작하는 trackball 을 openGL 을 이용하여 구현해 보았습니다.

[실행 환경]
Ubuntu 18.04
OpenGL version string: 2.1 Mesa 20.0.8

sudo apt-get install gcc
sudo apt-get install g++
sudo apt-get install freeglut3-dev
sudo apt-get install mesa-utils

[실행 방법]
mkdir build
cd build
g++ -o ../trackball ../trackball.cpp -lGL -lglut -lm -lGLU
../trackball.cpp

[조작 방법]
그냥 마우스로 드래그 시 rotate
shift 키를 누른 채로 드래그 시 translation
ctrl 키를 누른 채로 위로 드래그 시 zoom in, 아래로 드래그 시 zoom out
alt 키를 누른 채로 위로 드래그 시 dolly in, 아래로 드래그 시 dolly out
a 키를 한번 누르면 바로 show all

[구현 및 quaternion 연산 환경 설정]

먼저, 카메라의 위치를 x_0, y_0, z_0, 카메라의 up vector 를 camx, camy, camz, 바라보는 원점의 위치를 origin_x, origin_y, origin_z 로 표현하였습니다.

quaternion 연산을 구현하기 위하여 quaternion 이라는 구조체를 구현하였습니다. 4 개의 좌표는 float 4 개 w, x, y, z 로 표현되었습니다.

X2Q 함수는 x 축 방향으로 마우스 커서가 이동하였을 때 해당 회전을 quaternion 으로 변환하는 함수입니다. X 축 방향 이동은 y 축 기준으로 회전이며, 이 y 축이란 것은 camera 의 local coordinate 기준이므로 camx, camy, camz 을 normalize 하여 그 축을 기준으로 회전 변환을 주었습니다.

Y2Q 함수는 X2Q 함수와 비슷하게 y 축 방향으로 마우스 커서가 이동하였을 때 회전을 quaternion 으로 변환하는 함수입니다. X 축 기준으로 회전하는데, 이때 카메라의 local coordinate 기준 x 축이므로 y 축(camx, camy, camz)과 z 축(카메라위치 – 원점위치)를 cross product 하여 normalize 한 후 사용하였습니다.

Mul 함수는 quaternion 간 곱셈 연산을 구현하였고, Inv 함수는 quaternion 의 역함수를 구현하였습니다. Pos 함수는 x, y, z 좌표를 넣어주면 실수부가 0 인 pure imaginary quaternion 을 구현하는 함수입니다.

Translation 함수는 현재 position 을 나타내는 quaternion 과 x, y 값을 받아 camera local coordinate 기준으로 x, y 방향으로 translate 해주는 함수입니다. Dolly 함수는 Translation 함수와 비슷하지만, z 값을 받아 바라보는 원점에서 z 축 방향으로 멀어지거나 가깝게 해주는 함수입니다.

[구현 내용]

이를 이용하여 먼저 rotation 을 구현하였습니다. Sample 함수를 참고하여 마우스를 드래그 하면 x 의 변위에 따라 theta 가 측정되듯 y 축으로도 beta 를 측정하였고, 이를 이용하여 X2Q 함수와 Y2Q 함수에 넣어주고 Mul 연산을 취하여 회전 quaternion 을 완성하였습니다. 그 후 각각 카메라의 위치와 카메라의 up vector 에 대하여 q p q-1 회전연산을 실행하고, 각각의 좌표를 업데이트 해주었습니다.

다음으로 translation 을 구현하였습니다. Translation 시 카메라가 그대로 이동하는 효과를 주기 위해선 카메라가 움직인 벡터만큼 원점도 똑같이 움직여야 하며, up vector 은 움직이면 안 됩니다. 그래서 각각 카메라 위치 좌표와 원점 좌표를 quaternion 으로 변환하여 Translation 함수에 넣어주어 변환해 주었습니다.

다음으로는 zoom 구현입니다. Zoom 의 경우 viewing angle 을 키우거나 줄여야 하므로 y 방향의 마우스 움직임을 입력으로 받아 위로 가면 viewing angle 을 키우고, 아래로 가면 줄이는 방식으로 구현하였습니다. 여기서 viewing angle 이 180 도를 넘어가면 반대편으로 넘어가는 효과가 생겨 viewing angle 을 1 도와 179 도 사이로 제한하였습니다.

기본 구현 중 마지막인 dolly 구현입니다. Zoom 구현 방식과 비슷하게 y 방향의 마우스 움직임을 입력받아 위로 가면 원점 방향으로 카메라가 이동하고, 아래로 가면 원점 반대 방향으로 카메라가 이동하도록 구현하였습니다.

Extra credit 구현 중 show all 기능을 구현하였습니다. Show all 을 구현하기 위해선, 어느 상황에서든 카메라 방향은 돌리지 않은 채 물체의 모든 부분이 다 보이도록 움직이는 기능입니다. 이를 구현하기 위해, 카메라가 바라보는 벡터인 x_0 – origin_x, y_0 – origin_y, z_0 – origin_z 가 변하지 않도록 하며 바라보는 원점을 0, 0, 0 으로 이동시키고, 카메라의 위치도 이와 동일한 벡터만큼 움직였습니다. 그 후 카메라의 거리를 normalize 한 뒤 distance 를 3 으로 주어 구현하였습니다.

[구현하지 못한 것]

Extra credit 구현 중 Seek 기능은 구현하지 못하였습니다. 어떤 점을 화면에 찍었을 때 그 점으로부터 나오는 ray 를 만들어 물체에 닿을 때까지의 depth 를 측정해 보려 시도하였지만 잘 되지 않았습니다.