Report on Graphics Hw2
2013-11432 조성민

1. SETUP :

1. sudo apt-get install python3
2. sudo apt-get install python3-pip (if pip3 is not installed)
3. sudo pip3 install PyOpenGL

2. RUN:

1. python3 main.py

3. INSTRUCTIONS:

1. drag around with left click to rotate the trackball
2. drag around with right click to translate
3. press i or o to zoom out or in
4. press k or l to dolly out or in
5. press r to see all
6. press s to seek to where mouse is at

4. Implementation

저는 카메라 클래스를 구현했습니다. 카메라 클래스는 여러 정보를 가지고 있지만 가장 핵심적인 정보는 focus, 와 orientation 입니다. 카메라의 위치는 focus 에서 (0, 0, r) 이 orientation 만큼 rotation 된 곳만큼 간 곳에 있습니다. 쉽게 말하면 카메라는 focus 가 중심인 어떤 구 위에 붙어있고 orientation 이란 구 위에 카메라가 어디에 붙어 있느냐, 혹은 구가 얼마나 돌아가 있느냐를 나타낸다고 생각할 수 있습니다. 따라서 카메라의 위치는 항상 focus 와 orientation 으로부터 계산이 가능하고, 계산을 줄이기 위해 focus 나 orientation 이 바뀔때마다 카메라의 위치도 같이 계산해서 저장해둡니다. 카메라는 up 벡터 또한 가지고 있고 이는 (0, 1, 0) 을 orientation 으로 rotation 한 것입니다. 이 또한 orientation 이 바뀔 때마다 계산해서 가지고 있습니다.
glut 이 display 할때마다 camera 의 lookat() 을 호출하는데 그러면 gluLookAt() 을 통해 “카메라의 위치” 에서 “up vector” 를 가지고 “focus” 를 바라보게 됩니다.
“트랙볼” 이라는 것은 이해하기 쉽게 비유하면 카메라의 focus 가 중심이고 어딘가에 카메라가 붙어 있는 구라고 생각하면 편합니다. 트랙볼을 돌리게 되면 그 구의 orientation 이 바뀌게 되어서 카메라가 같이 돌아가게 됩니다. 즉 centre of rotation 이 카메라의 focus 가 됩니다. 카메라의 transition 이라는 것은 그 전체 구가 이동하는 것이라고 생각하면 좋습니다.
유저가 왼쪽 마우스 클릭으로 드래그 할 경우 바로 전의 마우스 좌표와 바로 지금 마우스 좌표가 트랙볼 상에 어떤 곳을 가르키는 벡터인지 계산합니다. 즉 구 위의 어느 점을 가르키는 좌표가 됩니다. 항상 트랙볼 중에 스크린 쪽을 향하고 있는 “반구”만 고려하므로 벡터의 z 좌표는 항상 0보다 크거나 같습니다. 트랙볼 바깥을 클릭하면 그 방향을 향하는 구의 가장 끝점이라고 생각합니다. 즉 (x, y) 가 그 방향을 향하는 구 위의 (x, y, 0) 이 됩니다. 구한 두 벡터로 부터 회전축과 회전각을 구할 수 있습니다. 구한 회전축은 구의 orientation 으로 한번 돌려줍니다. 그것이 “실제” 회전축이기 때문입니다 (구한 두 벡터를 미리 돌려서 “실제” 두 벡터를 구하고 그로부터 “실제” 회전축을 구했어도 됐지만 결국 같습니다). 구한 회전축과 회전각으로 구의 orientation 을 rotation 해줍니다. 이 때 카메라의 위치와 up vector 는 당연히 바뀌게 됩니다.
유저가 오른쪽 마우스 클릭으로 드래그 할 경우 바로 전의 마우스 좌표와 바로 지금의 마우스 좌표간의 차를 계산합니다. 그 후 카메라의 focus 를 오른쪽으로 x좌표의 차이만큼, 위쪽으로 y좌표의 차이만큼 이동하게 됩니다. 이 때 “오른쪽” 과 “위쪽” 이라는 것은 (1, 0, 0) 과 (0, 1, 0) 을 각각 카메라의 orientation 대로 rotation 한 것을 의미합니다. dolly in out 이라는 것도 결국 “앞쪽” 으로 특정 거리만큼 이동한것인데 이 때 “앞쪽” 이란 것도 (0, 0, -1) 을 카메라의 orientation 대로 rotation 한 것입니다. 이런식으로 이동하면 유저의 “직관” 대로 카메라가 움직이게 됩니다.

zoom in out 은 그냥 카메라의 field of view 를 조절해줍니다. 이 때 fov 가 1 보다 내려가거나 179 보다 높아지지 않도록 되어있습니다.
see all 을 하게 되면 카메라의 orientation 은 유지한 채로 focus 만 다시 원점으로 바꿔주고 field of view 를 원래 값으로 돌려 줍니다. 이렇게 하면 씬 전체를 다시 볼 수 있게 됩니다.
seek 은 다음과 같은 방식으로 작동합니다. 아이언맨을 그릴 때 입체도형들을 그릴 때 마다 그들의 주요 포인트 (예를 들어 큐브 라면 변들 위의 점들) 의 스크린 상의 픽셀 위치를 저장해둡니다 (by glProject()). 사용자가 seek 을 할 경우 마우스의 스크린 상 위치(깊이는 0)와 가장 가까운 포인트 (포인트는 깊이가 있음)를 고른 후 그 포인트의 3D 상 실제 위치를 구하고 (by glUnProject()) 그 곳으로 카메라의 focus 를 이동합니다. 자연스럽게 center of rotation 도 그 점이 됩니다. 예를들어 머리로 seek 한 후 rotation 을 해봤다가 발 쪽으로 seek 한후 rotation 을 해보길 바랍니다. 구심점이 바뀐것을 확인할 수 있습니다. 혹은 dolly out 과 translation 을 통해 아이언맨으로 부터 멀리 떨어진 후에 마우스를 아이언맨의 특정 부분에 가깝게 둔 후 s 를 눌러 보시길 바랍니다. 카메라가 그 곳으로 이동하는 것을 확인할 수 있습니다.

5. 그 외

seek 의 효과를 잘 확인하기 위해 아이언맨이 움직이는 것은 멈춰두었습니다.