

[Figure 1] 사용 매뉴얼

1. 구현 여부

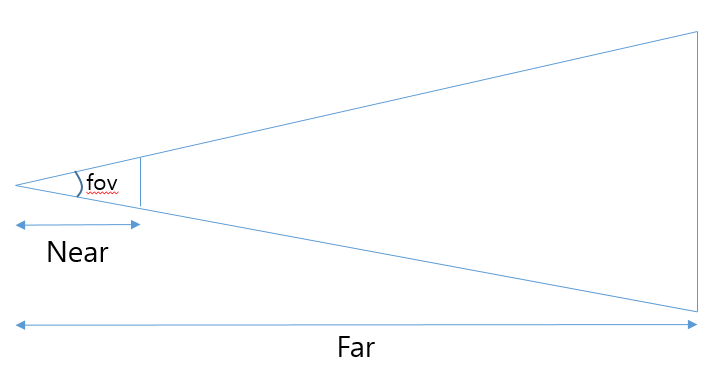
Rotation [ O ]  
Translate [ O ]  
Dolly in/out [ O ]  
Zoom in/out [ O ]  
Show all [ O ]  
Seek [ O ]

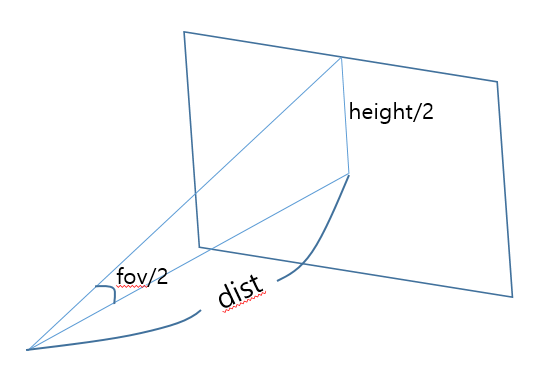
1. 실행 방법

g++ -o HW\_2 HW\_2.c –lm –lGL –lGLU –lglut

./HW\_2

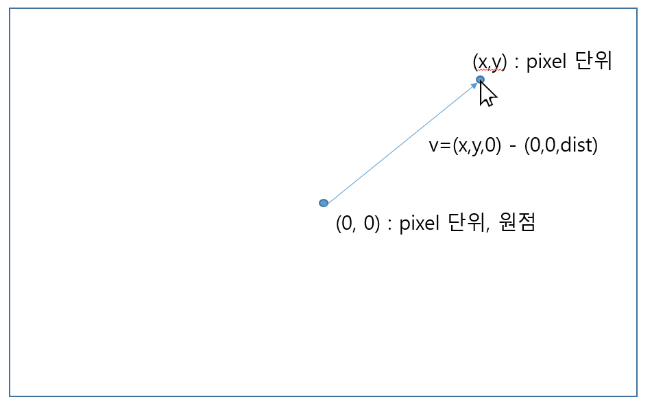
단, HW\_2.c 와 rhs\_math.h 가 같은 파일에 존재해야 한다.

1. 작동 원리
   1. Trackball 과의 접점을 구하는 방법



[Figure 1] frustum 단면도

[Figure 2]가상의 카메라와 viewport 사이의 거리 dist(pixel 단위)

[Figure 2]를 통해서 dist = (height/2) / tan(fov/2) 라는 것을 알 수 있다. 따라서 카메라에서 유저가 찍은 mouse pointer 까지의 방향 벡터를 [Figure 3]에 나온 벡터 v로 표현 가능하다. 

실제 구현에서 eye, ori 변수가 각각 카메라와 회전 중심의 좌표이다. 따라서 eye + k \* v 벡터가 ori를 중심으로 하며 반지름이 R 인 구 위에 만나는 점을 구하면 trackball과의 접점을 구할 수 있다. 만약 접점이 없다면 trackball로 향하는 접선을 계산하여 적용하였다.

* 1. Quaternion 구하기

과정 A를 통해 점을 구할 수 있다. 이번에 찍힌 점과 직전에 찍혔던 점과의 관계를 이용하여 회전 축과 각도를 구할 수 있다. 이를 quaternion으로 변환하며 누적 시켜서 최종적으로 gluLookAt 함수에 인자를 넣어줄 때 회전시킨 값을 적용하였다.

* 1. Seek 구현 방법

glReadPixels 란 함수를 이용하였다. 해당 픽셀의 z-buffer를 통해 실제 z 값을 계산해 낼 수 있다. 과정 A를 통해 camera에서 나오는 방향 벡터를 알고 z 값을 알기 때문에 어느 지점에 물체의 표면이 놓여있는 지 알 수 있다. 그 점을 P라고 구했으면, P-ori 의 벡터 만큼 ori, eye 를 translate 시켜서 회전 중심이 찍은 물체의 표면이도록 구현했다.

* 1. Show All 구현 방법

물체의 모든 점이 화면에 표현되어야 한다. 이를 이용하여 최소로 필요한 높이와 너비를 알 수 있다. 5가지 정다면체마다 물체의 높이를 계산할 수 있기 때문에 이를 이용하여 최소한 (0,0,30) 점이 보이면 된다는 것을 계산해 내었다(해당 점은 rotation, translate가 없을 때의 값이다.) 카메라의 좌표와 필요한 높이를 알기 때문에 [Figure 2]와 같은 방법을 이용하여 얼마나 카메라를 뒤로 빼야 하는 지 알 수 있다. 이와 같은 방법으로 Show All을 구현하였다.

* 1. Translate, Dolly in/out, Zoom in/out 구현 방법

Translate는 A, S, D, W 키를 눌렀을 때, ori와 eye를 같은 방향으로 동시에 움직이도록 구현했다.

Dolly in/out은 카메라의 좌표와 회전 중심의 좌표 사이의 거리를 조절하여 구현했다. 최소 거리와 최대 거리에 제한을 두었다.

Zoom in/out은 fov 값을 조절하였다. 최소 값과 최대 값을 설정해 놓았다.