Graphics HW2 Report

컴퓨터공학부 2013-11425 이창영

**1. 사용 언어**

C++

**2. 컴파일 방법**

g++ hw2.cpp -o hw2 -lm -lGL -lGLU – lglut

**3. 실행 방법**

./hw2

**4. 사용한 라이브러리**

함께 첨부되어 있는 “quaternion.h”

quaternion 관련 연산을 해주는 github에서 가져온 소스.

“quaternion.h”에서 이번 과제에서 필요한 multiply, magnitude, normalize, conjugate를 제외한 불필요한 부분은 모두 삭제하였다.

출처 – <https://github.com/adafruit/Adafruit_BNO055/tree/master/utility/quaternion.h>

**5. 구현한 것**

1) virtual trackball rotate

2) translate screen

3) zoom in/out

4) dolly in/out

**6. 구현하지 못한 것**

(extra credits)

1) Show all

2) Seek

**7. 테스트 방법**

1) virtual trackball rotate

마우스 왼쪽 버튼을 클릭하여 드래그 하면 virtual trackball 방식으로 rotate하는 모습을 볼 수 있다.

2) translate screen

키보드 t버튼을 누른 상태로 유지하며 마우스를 드래그 하면 마우스 방향으로 전체가 translate된다.

3) zoom in/out

키보드 z버튼을 누른 상태로 방향키 위 버튼을 누르면 zoom in 되고, 방향키 아래 버튼을 누르면 zoom out 된다.

4) dolly in/out

키보드 d버튼을 누른 상태로 방향키 위 버튼을 누르면 dolly in 되고, 방향키 아래 버튼을 누르면 dolly out 된다.

**8. 구현 내용, 방법**

1) 가정

rotate, zoom in/out, dolly in/out의 중심은 항상 화면의 중심이다.

2) 초기 설정(실행 했을 때)

width: 1200, height: 800의 창 크기를 가지고 있다.

물체의 중심은 (0.0, 0.0, 0.0)에 위치해 있다.

카메라는 (0.0, 0.0, 350.0)에 위치해 있다.

카메라는 (0.0, 0.0, 0.0)을 바라보고 있다.

카메라의 up벡터는 (0.0, 1.0, 0.0).

3) virtual trackball rotate

I) 화면의 가로와 세로 중 작은 쪽의 1/2의 0.85배를 반지름(r)으로 하는 구가 (0.0, 0.0, 0.0)을 중심으로 있다고 설정하였다.

II) 마우스를 드래그 할 때 이전 마우스 포인터의 위치와 나중 마우스 포인터의 위치를 구 위에서의 좌표로 변환하였다.

* 마우스 포인터의 위치를 (a, b)라고 할 때, 좌표(x, y)는 (a – width/2, height/2 – b)가 된다.
* x^2 + y^2 <= r^2인 경우 x^2 + y^2 + z^2 = r^2을 이용해서 z좌표를 구하였다.
* X^2 + y^2 > r^2인 경우 원점에서 그 좌표를 이은 직선과 반지름이 r인 원의 교점을 마우스가 선택한 (x, y)좌표로 정했고 z좌표는 0으로 하였다.

III) 이렇게 구한 구 위에서의 좌표는 처음 아무런 회전이 없는 경우엔 그대로 사용해도 되지만 초기 상태가 아닌 회전이 이미 있는 상태에선 이 좌표를 그대로 사용하면 안되고 그만큼 회전된 값을 사용해야 한다.

* 이를 위해 초기 상태로 부터의 현재 회전 상태를 알 수 있도록 글로벌 변수로 quaternion변수 rq, rq^(-1) 두개를 두었다.
* 구한 구 위에서의 좌표를 quaternion p로 변환한 뒤, rq \* p \* rq^(-1)하여 회전 까지 적용된 좌표를 구하였다.

IV) III을 통해 구한 두 좌표를 원점으로부터의 벡터로 바꾼 뒤 두 벡터 사이의 각도와 기준 축을 구하였다. (atan2 사용)

V) 기준 축과 사이의 각도를 가지고 이 회전의 반대로 회전하는 것을 나타내는 quarternion q으로 변환하였다. (돌리는 방향과 반대 방향으로 카메라가 회전해야 하므로)

VI) 이 q을 가지고 현재 카메라의 위치와 카메라의 up 벡터를 회전시켜 준다. 이 때 아까 설정했던 rq와 rq^(-1)에도 이 q \* rq, rq^(-1) \* q(-1)를해서 업데이트 해 주었다.

4) translate screen

I) 위 트랙볼과 마찬가지로 마우스를 드래그 할 때 이전 마우스 포인터의 위치와 나중 마우스 포인터의 위치를 가지고 이전 좌표와 나중 좌표를 구한다.

* 마우스 포인터의 위치를 (a, b)라고 할 때, 좌표(x, y)는 (a – width/2, height/2 – b)가 된다.
* z좌표는 0으로 두었다.

II) 두 좌표 사이의 벡터를 구한뒤 이를 quaternion으로 변환하고 위 트랙볼에서 업데이트 해 주덨던 rq, rq^(-1)을 앞뒤에 곱하여 현재 회전 상태에 맞는 벡터를 구하였다.

III) 화면 전체를 translate하는 코드를 display 첫부분에 translate(a, b, c)로 둔 뒤 II에서 구한 벡터에 맞게 a, b, c를 변화시켰다.

5) zoom in/out

I) gluperspecrive부분의 시야 각을 좁히면 zoom in이 되고 시야 각을 넓히면 zoom out이 된다.

6) dolly in/out

I) 원점에서 카메라 방향으로 향하는 벡터를 구한뒤 그것의 normalized 벡터를 구했다.

그 벡터 방향으로 카메라를 보내면 dolly out 반대 방향으로 보내면 dolly in이 된다.