**기초 컴퓨터 그래픽스**

**HW1 README**

20171300 지용환

**1. [환경 명세]**

본인 프로그램의 실제 구동 환경을 명시할 것 (OS, CPU, GPU, Complier 등)

Windows 10 64bit, i7-6700, GTX 1060, Visual Studio 2022 Release x64

**2. [요구사항]**

(a) 윈도우 화면

- 구현 여부 : 예

- 작동 확인 방법 : 프로그램을 실행하면 윈도우가 화면에 뜬다.

- 구현 방법 : 

이 코드는 주어진 파일에 이미 있던 코드로, 수정없이 그냥 750,750사이즈의 윈도우가 나오게 해준다.

(b) 선분 그리기

구현: o

직사각형이(가) 표시된 사진

자동 생성된 설명

방법:

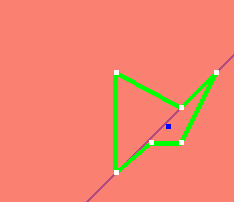
텍스트이(가) 표시된 사진

자동 생성된 설명(수정한 부분은 gllinewidth)

위의 draw\_line의 수정을 통해 구현했다. gllineWidth로 선분의 굵기를 바꾸어주고, 좌표(qx,qy,px,py)를 (-0.1,1.0) ~ (1.0,1.0) 로 초기화 해주어 (초기화는 initialize\_render부분에서 그냥 value값 설정) 그 선분을 glBegin ~~~glEnd로 그리게 설정하였다. 선분 각 끝 꼭짓점을 표시하는 부분은 딱히 수정할 필요가 없어 수정하지 않았다.

(c) 비대칭 다각형 그리기

구현 : o



방법:

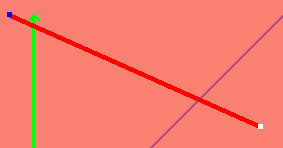
위의 선분과 비슷하게

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명 initialize\_render부분에서 좌표 value값을 적절히 설정해 주었다. 예시 코드와는 좌표, 모양, 6각형으로 다르게 만들어 주었다. Draw\_object에서 이 다각형을 그리는데 display함수에서 먼저 선분을 그리고->다각형을 그리기에 미리 설정한 linewidth=5가 변하지 않아 이 다각형을 그리는데는 아무 부분도 수정하지 않았다. 다각형을 그리기 위해선 glLineLoop를 사용한다.

(d) 선분의 회전(파란색 꼭짓점 기준)

구현 :o

(휠로 회전한 선분. 위의 선분과 비교)

방법:

Rotation matrix와 transform matrix를 사용해 구현하였다. 딱히 복잡한 것을 만드는 것이 아니아 본 프로젝트에선 어떠한 포인터도 사용하지 않았기에, 모든 matrix를 배열로 구현을 했다. 이때 행렬 곱은 아주 자주 사용하니 함수로 만들었는데, 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

P는 미리 정의한 구조체로서, member변수에 x,y좌표가 존재한다. 함수의 parameter중 matrix는 당연히 아핀변환행렬이고, x,y는 변환하고자 하는 x,y좌표이다. 즉 변환하고자하는 x,y를 이 함수내에서 새로운 배열로 만들어, 행렬곱을 하고 그 만들어진 행렬의 (변환된) x,y좌표를 그대로 P구조체 변수로 반환하게 해주었다.

이제 휠 기능을 설명하겠다.

휠은 mousepress에서 인식이 가능하고, button=3 또는 4로 값이 들어온다. 이것을 인식해, 회전을 하게 했는데 전역변수 rotate는 휠을 돌리면 1이 되며, 이때 지역변수 radius(초깃값0)를 수정시킨다.

텍스트이(가) 표시된 사진

자동 생성된 설명(휠을 위로 돌리면 실행되는 파트. Win\_z는 나중에 설명). 수정된 값을 가지고 rotate=1이 되면

텍스트이(가) 표시된 사진

자동 생성된 설명다음과 같은 코드를 실행한다.

~~~\_matrix는 아핀변환 행렬이다. 모두 지역변수로 구현을 해놨기에 설명은 생략하겠다. 먼저 파란색점 중심이기에 (좌표는 px,py)을 원점으로 이동시키고 (이동행렬 이용) 그리고 휠을 돌림으로서 바뀐 radian을 이용해 실제 라디안 값으로 바꿔주고 그걸로 cos, sin을 만든후, 회전행렬의 value에 할당시켜준다. 그 이후 회전을 시키고, 다시 px,py만큼 이동시켜주는 방법을 선택했다. 당연히 흰색점이 회전하는 것이기에 qx,qy에 계산된 value를 할당시켜주고 다시 그리게 gluatpostredisplay를 통해 구현을 하였다. 계속 도는 것을 막기 위해 이 부분을 실행한 이후 roate는 다시 0이 되어 휠 입력을 기다리게 된다.

(e) 쉬프트, 왼쪽 마우스로 파란색 꼭짓점의 이동(picking)

구현 :o

다각형이(가) 표시된 사진

자동 생성된 설명(위 사진과 비교)

방법:

먼저 이 기능에는 어떠한 아핀 변환을 하지 않았다. 그냥, 쉬프트와 마우스 왼쪽을 인식하고 마우스 좌표만큼 px,py를 이동시키게 해주었다. 가장먼저 약간의 좌표 변환을 설명하겠다. Mousepress, mousemove같은 곳에서 parameter로 들어오는 마우스의 위치는 윈도우 좌표계 기준이므로 opengl의 좌표계와 다르므로 수정이 필요했다. 물론 행렬곱 (opengl중심축의 음수 만큼 이동행렬 -> 그후 x축 반사행렬으로 같은방법 가능)으로도 구현이 가능하지만, 그냥 수식으로도 가능하기에 간단히 구현했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

먼저 x,y는 윈도우 좌표계의 좌표고, real\_x와 real\_y는 opengl좌표계이다. 행렬의 곱과정이랑 당연히 의미가 같고 윈도우 250픽셀이 길이 1이므로 250까지 나누어주었다. 자주 사용하지만 이제 이에 대한 설명은 생략하겠다.

본격적으로

쉬프트 ,alt, ctrl같은 스페셜 키는mousepress함수 안에서

를 통해인식을 했고, 동시 인식 방법은 다음과 같다. Shift\_on은 초깃값0인 전역변수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Glut\_down은 마우스의 클릭을 인식 그리고 button은 뭘 눌렀는지를 인식한다. 또한 specialKey로써 쉬프트가 눌린 것을 확인, 모두다 통과를 한다면 windown좌표계를 opengl좌표계로 변환하여 현재 마우스가 파란색 꼭짓점 주위에 잇는지 확인하게 해주었다. 오차는 +-0.01f만큼 주었다. 파란색 꼭짓점근처에 있다면 shift\_on을 1로 만들어 주어, mousemove함수에서 다음과 같은 기능을 하도록 하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명(mousemove)

mousemove함수에서도 좌표 변환을 하였고, 계속 쉬프트가 눌린 것을 감지 하기위해 여기서도 specialKey를 통해 shift를 눌러야지만 작동하게 해주었다. 즉, 도중에 쉬프트키를 때면 이 if문으로 들어오지 않는다. 아무튼간, shift를 누른상태라면 들어와 좌표변환을하고 px,py를 변환된 좌표로 이동시켜 picking을 구현했다.

(f) alt + 오른쪽 마우스 버튼으로 다각형의 이동 (translation)

구현:o

도표이(가) 표시된 사진

자동 생성된 설명

방법: picking에서 구현한 것과 비슷하게 좌표변환을 이용했고, 여기엔 이동행렬의 곱을 이용해 구현해 주었다. 먼저 인식 부분은 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

오른쪽 마우스 버튼과 alt키 동시 수행시 들어오며 alt\_on은 전역변수, 그리고 at\_alt\_po는 마우스의 움직임(dx,dy)을 파악하기 위해 계속 value를 바꾸는 전역변수이다. mousemove함수는 다음과 같다. 앞서 말했듯이 alt를 계속 누르고 있어야 이 if문으로 들어올수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

At\_alt\_po의 좌표와 마우스의 현재 좌표의 차를 이용해 move\_x,move\_y즉 dx,dy의 값을 구하고 당연히 다음 움직임도 파악해야 하기에 at\_alt\_po값을 현재 마우스 좌표로 변경해준다. move값을 이용해 이동행렬의 값을 수정, 그리고 6각형의 모든 꼭짓점에 대해 이동행렬을 곱해주어, 꼭짓점의 모든 좌표를 수정해주면 된다. 추후, 사용하는 무게중심또한 이동한 좌표에 맞게 계속 계산을 통해 수정한 이후, glutpostredisplay로 새롭게 이미지를 드로잉시켜주면 된다.

(g)ctrl + 마우스 오른쪽 버튼으로 도형의 확대, 축소 (scaling)

구현: o

차트이(가) 표시된 사진

자동 생성된 설명

Ctrl과 마우스 클릭 인식은 다음과 같다.

차트이(가) 표시된 사진

자동 생성된 설명

scaling기능에서 도형의 확대와 축소는 오직 마우스의 왼쪽, 오른쪽 이동만 파악하면 되기에 at\_ctrl\_po\_x만 전역변수로 설정해 마우스이동 변화값, 방향을 계산했고, ctrl\_on은 초깃값0인 전역변수이다. Ctrl을 누른 상태로 movsemove함수 부분은 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

마우스 왼쪽이동 (축소) 오른쪽 이동(확대)를 따로 구분했지만 비슷하기에 한부분만 설명하겠다. (지금보니 한번에 구현도 가능하겠지만..) 왼쪽 이동시에 마우스 이동 변화값을 계산해서 size를 계산하고 scaling\_matirx에 그 값을 넣어 수정해 준다. 주의할 것은 1-size인데, 그냥 size값을 대입할시, size값이 워낙에 작기에 도형이 한번에 없어지므로 주의해야된다. 이후에 6각형의 모든 꼭짓점을 무게중심만큼 이동행렬 곱 (무게중심이 원점에 오게)-> 그후 확대(혹은 축소)행렬 곱 -> 다시 원상복구 위해 무게중심만큼 이동행렬을 곱해주는 과정이다. 즉 3번의 행렬 곱을 하고, 그 값을 꼭짓점에 새롭게 할당해주면 확대와 축소의 구현이 완료된다. 특수키입력의 모든 확인변수들은

텍스트이(가) 표시된 사진

자동 생성된 설명

mousepress에서 마우스를 땔시 0이되고 또한

텍스트이(가) 표시된 사진

자동 생성된 설명

mousemove에서도 특수키를 때는 경우를 else로 만들어 0이되게 초기화 시켰다. 이로써, 특수키의 입력을 계속 인식 가능하게 하였다.

(h) 추가 구현 = y=x기준으로 shering그리고 y=x기준 반사된 도형이 그려지게

구현 : o

기능 : z를 누르면, 화면 사이즈가 750,750으로 초기화 되며, 기존 보라색 선(y=x함수)의 우측편에 새로운 도형이 그려진다. 이때, z를 누른상태로 마우스 휠을 돌리면 shearing

X를 누르면 y=x에 대칭적인 도형이 그려진다. 즉, z를 누른상태로, 휠과 x를 누르며 결과 를 확인 가능하다.

차트이(가) 표시된 사진

자동 생성된 설명 (z입력시에 생기는 도형. Shearing을 했던 말던, 항상 z를 누르면 이 위치에서 초기화된 도형이 생성되며, z에서 손을때면 보이지 않는다. )

차트이(가) 표시된 사진

자동 생성된 설명(휠 굴리면, y=x함수 위에서 shearing)

차트이(가) 표시된 사진

자동 생성된 설명(청록색의 도형은 x를 누르면 보이는 대칭 도형이다. X를 누른상태로 휠을 돌리는 것도가능. 당연히 반대도 가능하다. 다만 x에서 손을때면 청록색도형은 안보임)

먼저 z입력시에 도형을 그리는 방법은 다음과 같다. 이때 win\_z는 초깃값 0인 전역변수이다.

텍스트이(가) 표시된 사진

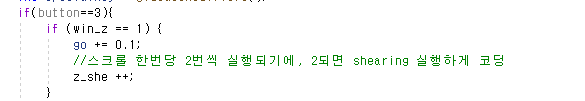
자동 생성된 설명

keyboard콜백 함수에서 z를 입력시에 win\_z를 1로 만들며, 5각형을 초기화하며(initial\_5rec은 5각형의 모든 좌표value를 초기화 해주는 함수이다.) 그림을 그린다. 이때, z를 누르면 계속 keyborad함수를 호출하므로, win\_z변수를 이용해 처음 z를 누르면 win\_z값을 변경하고 그 한번만 초기화를 하도록 해였다. 이 5각형은 darw\_object에서 win\_z==1일시 그냥 그리게 설정해주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이제 휠로써, shearing한 부분을 설명하겠다.



앞서 캡쳐된 코드중 일부인데, win\_z==1 즉 z가 눌린 상태일 때, 휠을 누른다면 지역변수 go의 증가, 전역변수 z\_she가 증가하게 하였다. 확인을 해보니, 휠을 한번 돌릴때마다 mousepress함수가 2번씩 호출되어, 아래처럼 z\_she가 2가 되어야 if문에 들어갈수 있도록 했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

아무튼, 휠을 인식에 z\_she가 2가 된다면, 먼저 y=x를 정확히 y과 동일하게 만들기위해 45도 회전을 한 이후, 회전된 좌표에 y좌표 shearing을 해주었고, 다시 -45도 회전을 통해 y=x축 위에서의 shearing을 구현했다. 실제로 she\_matrix의 2행 1열의 값이 go가 되고, 3번의 행렬곱을 통해 구현했다.

추가적으로 x버튼을 누른다면

텍스트이(가) 표시된 사진

자동 생성된 설명

Win\_x를 1이되게 초기화 했고, 이는 draw\_object에서 win\_x==1일 때 대칭적인 그림을 그리도록 했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Shearing과 유사하게 먼저 45도 회전을 한 이후, 정확히 y축 대칭이동, 그리고 다시 -45도 회전을 해 y=x대칭 그림을 그렸다. 이때 대칭되어 보이는 도형은 실제 변수가 아니므로, 그냥 계산되는 s\_p의 x,y좌표를 이용해 그렸다.

당연히, x혹은 z키에서 손을 땐다면 그림은 안보여야 하는데, 키보드up은 새로운 callback함수를 추가했고,



이를 바탕으로 keyup콜백함수를 다음과 같이 구현했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

간단히, z혹은 x에서 손을때면, 변수를 0으로 만들어 그림을 그리게 해주는 것이다. 그림을 그리는 모든 부분은 draw\_object에서 if문으로 win\_z,x를 구분하여 그렸기에 0이 된다면, 즉 키up을 한다면 그림은 보이지 않게 된다.