**기초 컴퓨터 그래픽스**

**HW1 README**

학번 : 20191583 이름 : 김태곤

**1. [환경 명세]**

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

* ex) Windows 11 64bit, i5-12600KF, RTX 3080, Visual Studio 2022 Release x64

**2. [요구사항]**

(a) 윈도우 화면

- 구현 여부 : 예

- 작동 확인 방법 : 프로그램을 실행하면 윈도우가 화면에 뜬다.

- 구현 방법 : glutInitWindowSize(750,750)

glutInitWindowSize를 이용해 초기 윈도우 화면의 크기를 가로, 세로 각각 750 픽셀로 지정하였다.

(b) 선분 그리기

- 구현 여부 : 예

- 작동 확인 방법 : 윈도우 화면에 한 꼭지점은 파란색, 다른 꼭지점은 흰색인 적절한 크기의 선분이 표시된다.

- 구현 방법 : px=-0.7f, py=-0.5f, qx = 0.0f, py = -1.0f

초기 선분의 꼭지점 좌표를 초기화해주었다. 이를 바탕으로 콜백함수 안에 glutDisplayFunc이 display를 실행시켜 draw\_line을 이용해 선분을 그렸다.

(c) 비대칭 다각형 그리기

- 구현 여부 : 예

- 작동 확인 방법 : 윈도우 화면에 흰색 점으로 이루어진 적절한 크기의 비대칭 다각형과 파란색 점의 무게 중심이 표시된다.

- 구현 방법 : draw\_axes() 함수에서 기존에 선언되어 있던 glVertex2f의 인수값을 수정해주었다.

(d) 선분 회전시키기

- 구현 여부 : 예

- 작동 확인 방법 : 마우스 휠을 위쪽으로 스크롤하면 파란색 점을 중심으로 선분이 회전한다. 아래로 스크롤하면 반대편 방향으로 회전한다.

- 구현 방법 :

마우스 휠의 입력을 받고 적용시키기 위해 콜백 함수에 glutMouseWheelFunc(mouseWheel);을 추가하였다. mouseWheel(int button, int dir, int x, int y)함수는 다음과 같이 정의하였다.

마우스 휠의 입력 방향을 나타내는 dir이 0보다 크면 rotation\_angle\_in\_degree는 양수값 1.0f, 아니면 음수값 -1.0f을 가진다. 이 각도를 라디안으로 바꾼 후 간단한 식을 위해 cos값과 sin값을 구해주었다. Rotation을 위해 T(px,py)R(theta)T(-px,-py) 방식을 이용하였다. 먼저 qx, qy의 값에 각각 px,py만큼 빼주고, qx = qx\*cos - py\*sin, qy = qx\*sin + qy\*cos을 해주었다. 이후 처음 translation시킨 만큼 반대로 translation시켜주어(qx,qy에 각각 px,py 더해주어) 회전을 완료하였다. mouseWheel 함수 마지막에는 glutPostRedisplay()함수를 적용시켜 즉각적으로 적용 가능하게 하였다.

(e) 파란색 꼭지점 이동

- 구현 여부 : 예

- 작동 확인 방법 : SHIFT 키를 누른 상태로 파란색 꼭지점을 좌클릭하여 이동시킨다.

- 구현 방법 : 키 입력과 마우스 입력, 마우스의 이동이 동시에 필요하므로 callback함수에서 glutMouseFunc(mousepress) 와 glutMotionFunc(mousemove)를 이용하였다.

“if (button == GLUT\_LEFT\_BUTTON && glutGetModifiers() == GLUT\_ACTIVE\_SHIFT &&state == GLUT\_DOWN)” 먼저 mousepress에서 SHIFT키 입력과 마우스의 입력이 들어오는 것을 확인하였다. 먼저 윈도우 창 기준으로 입력 받는 좌표를 (x-375)/250, -(y-375)/250을 통해 화면에 그려진 xy축 기준 좌표로 바꾸었다. 입력된 마우스 포인터의 위치가 일정 범위(0.03) 안에 들어올 경우 leftbuttonpressed를 1로 만들어 움직일 준비를 하였다. 이후 mousemove 함수에서 leftbuttonpressed가 1이고, SHIFT키가 눌려 있을 때 px+=dx, py+=dy를 통해 translation을 구현하였다.

(f) 다각형 위치 이동

- 구현 여부 : 예

- 작동 확인 방법 : ALT 키를 누른 상태로 화면을 우클릭하여 다각형을 이동시킨다.

- 구현 방법 : 키 입력과 마우스 입력, 마우스의 이동이 동시에 필요하므로 callback함수에서 glutMouseFunc(mousepress) 와 glutMotionFunc(mousemove)를 이용하였다.

if (button == GLUT\_RIGHT\_BUTTON && glutGetModifiers() == GLUT\_ACTIVE\_ALT && state == GLUT\_DOWN)을 이용하여 ALT키를 입력과 마우스의 입력이 들어오는 것을 확인하였다. 조건에 맞는 입력이 이루어진 경우 rightbuttonpressed를 1로 만들어 움직일 준비를 하였다. 이후 mousemove 함수에서 rightbuttonpressed가 1이고, ALT키가 눌려 있을 때 과제e에서 구한 방식을 다각형에 적용해 다각형을 translation시켰다. 각 꼭지점마다 object[i][0]+=dx, object[i][1]+=dy를 해주었고, 각 좌표의 x,y값을 각각 더해 전체 꼭지점의 개수로 나누어 주어 object\_center의 값을 구하였다.(object\_center의 값을 구하는 것은 과제 c에 구현된 것과 동일하다.)

(g) 다각형 크기 변환

- 구현 여부 : 예

- 작동 확인 방법 : CTRL 키를 누른 상태로 화면을 우클릭하여 다각형을 조절한다.

- 구현 방법 : 키 입력과 마우스 입력, 마우스의 이동이 동시에 필요하므로 callback함수에서 glutMouseFunc(mousepress) 와 glutMotionFunc(mousemove)를 이용하였다.

if (button == GLUT\_RIGHT\_BUTTON && glutGetModifiers() == GLUT\_ACTIVE\_CTRL && state == GLUT\_DOWN)을 이용하여 CTRL키를 입력과 마우스의 입력이 들어오는 것을 확인하였다. 조건에 맞는 입력이 이루어진 경우 center\_selected를 1로 만들어 움직일 준비를 하였다. scaling을 하기 위해 cx = object\_center\_x - object[i][0], cy = object\_center\_y - object[i][1]를 적용하여 다각형의 중심을 기준으로 크기 변환이 가능하게 하였다. 이후 object[i][0] = object\_center\_x - cx \* (1.0f+dx), object[i][1] = object\_center\_y - cy \* (1.0f+dx) 적용하여 마우스의 x축 이동에 따라 scale의 변화를 주었다.

(h) 추가 구현

추가구현은 shearing과 Reflection 총 2가지를 진행하였다.

Shearing은 다각형의 원점 좌표의 x축을 기준으로 진행하였다.

Reflection은 주어진 선분 (pxpy)(qxqy)를 기준으로 다각형을 대칭이동 시켰다.

h-1) 다각형 Shearing

- 작동 확인 방법 : Shift 키를 누른 상태로 화면을 우클릭하여 다각형을 조절한다.

- 구현 방법 : Shift키의 인식과 마우스의 인식은 앞의 과제 e,f,g와 동일하다.

Shearing을 하기 위해 먼저 다각형의 각 꼭지점을 다각형의 중심만큼 이동시켰다. (object[i][0] -= object\_center\_x, object[i][1] -= object\_center\_y) x축 기준으로 Shearing이므로 다각형의 y좌표는 그대로 두고 x좌표에 dx\*object[i][1]을 더해 기울기를 주었다.(object[i][0] = object[i][0] + dx\*object[i][1]) 이후 처음 이동시켰던 만큼 반대로 이동시켜 원위치를 시켜주었다. 다각형의 중심은 과제f에서 구현한 것과 동일하게 구현하였다.

h-1) 주어진 선분 기준 다각형의 대칭이동

- 작동 확인 방법 : 스페이스바를 누르면 화면에 표시된 선분 기준 다각형이 대칭이동한다.

- 구현 방법 : 스페이스바의 인식은 일반 키보드의 인식이므로 keyboard함수의 switch문에서 처리하였다. 스페이스바가 인식되면 먼저 math.h의 atan을 이용해 선분의 각도를 구하였다.(atan(qy-py)/(qx-px)) 선분을 기준으로 대칭이동을 시키기 위해서는 T(px,py)R(theta)R\*R(-theta)T(-px,-py)를 해주어야 한다. -px,-py Translation은 과제f의 이동 방식에서 +=을 -=으로 구현하였다. 선분의 각도에 대해 cos값과 sin값을 구하고 x = object[i][0] \* c - object[i][1] \* s, y = object[i][0] \* s + object[i][1] \* c를 해주어 다각형의 좌표를 주어진 축 기준으로 이동한 선분 기준으로 맞추었다. 이후 object[i][1] = -object[i][1]를 통해 대칭이동 시켰다. 원상 복구의 과정인 T(px,py)R(theta)는 위 과정의 반대로 진행하였다.