**기초 컴퓨터 그래픽스**

**HW1 README**

학번 이름 20171664 이상윤

**1. [환경 명세]**

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

* Windows 10 64bit
* AMD Ryzen 5 3600 6-Core Processor 3.59 GHz
* RTX 2060 SUPER/PCIe/SSE2
* Visual Studio 2022 Release x64

**2. [요구사항]**

**- 아래의 내용은 예시입니다. 확인 가능하도록 자유롭게 작성하시면 됩니다.**

(a) 윈도우 화면

main 함수의 glutInitWindowSize(750, 750) 함수 호출 부분에서 가로, 세로 각각 750 픽셀 크기의 윈도우가 화면에 뜨도록 skeleton code에 기본적으로 작성되어 있다.

(b) 선분 그리기

void initialize\_renderer(void) 함수를 통해 초기 도형들의 크기를 설정할 수 있다.

(px, py)는 파란점, (qx, qy)는 흰 점을 나타낸다.

이 값을 px = -1.0f, py = 0.20f, qx = -0.6f, qy = 0.20f 와 같이 변경함으로서 예제 프로그램과는 다르게 설정하였다.

(c) 비대칭 다각형 그리기

(b)에서와 마찬가지로 void initialize\_renderer(void) 함수를 변경함으로서 예제 프로그램과는 다르게 설정하였다.

설정한 값은 아래와 같다.

object[0][0] = sq\_cx + sq\_side;

object[0][1] = sq\_cy + sq\_side;

object[1][0] = sq\_cx;

object[1][1] = sq\_cy + 1.5 \* sq\_side;

object[2][0] = sq\_cx - sq\_side;

object[2][1] = sq\_cy + sq\_side;

object[3][0] = sq\_cx - sq\_side;

object[3][1] = sq\_cy - sq\_side;

object[4][0] = sq\_cx + 0.2 \* sq\_side;

object[4][1] = sq\_cy - 0.8 \* sq\_side;

object[5][0] = sq\_cx + sq\_side;

object[5][1] = sq\_cy - sq\_side;

(d) 꼭지점 회전

마우스 휠이 아래로 굴러갈 때 마다 콜백으로 등록한 void mousepress(int button, int state, int x, int y) 함수가 호출된다.

그때의 button 값은 MOUSE\_WHEELED 와 같다.

휠이 위로 굴러갈때 button 값은 MOUSE\_WHEELED - 1 이다.

그리고 회전하는 정도는 void initialize\_renderer(void) 함수에서 설정한 rotation\_angle\_in\_degree = 1.0f 값을 기준으로 한다.

회전을 위해선 원점으로 이동하고, 회전을 시킨다음, 다시 원래의 위치로 원위치 시키는 아핀 변환 과정들이 필요하다.

휠이 아래로 굴러갈때의 코드는 아래와 같다.

//원점 이동

float delx = -px;

float dely = -py;

float tempqx = qx + delx;

float tempqy = qy + dely;

//회전

tempqx = tempqx \* cos(-rotation\_angle\_in\_degree \* TO\_RADIAN)

- tempqy \* sin(-rotation\_angle\_in\_degree \* TO\_RADIAN);

tempqy = tempqx \* sin(-rotation\_angle\_in\_degree \* TO\_RADIAN)

+ tempqy \* cos(-rotation\_angle\_in\_degree \* TO\_RADIAN);

//원위치

qx = tempqx - delx;

qy = tempqy - dely;

glutPostRedisplay();

(e) 파란색 꼭지점 이동

마우스를 클릭하면 void mousepress(int button, int state, int x, int y) 함수가 호출된다.

이때 state는 마우스를 눌렀을때 GLUT\_DOWN, 마우스를 땔때는 GLUT\_UP 이다.

왼쪽 마우스를 눌렀다면 전역적으로 선언된 leftbuttonpressed 를 참으로, 뗄때는 거짓으로 바꾼다.

왼쪽 마우스를 눌렀다면 전역적으로 선언된 rightbuttonpressed 를 참으로, 뗄때는 거짓으로 바꾼다.

그리고 마우스를 이동하면 void mousemove(int x, int y) 함수가 호출된다.

마우스의 처음 클릭 위치를 기억하기 위해 전역적으로 int prevx, prevy 를 선언한다.

또한 glutGetModifiers() 함수는 shift키를 누를때는 GLUT\_ACTIVE\_SHIFT, ctrl를 누를때는 GLUT\_ACTIVE\_CTRL, alt키를 누를때는 GLUT\_ACTIVE\_ALT 를 반환한다.

또한 mousemove() 함수의 인자로는 윈도우 좌표계로 전달된다.

전달된 좌표(x, y)와 (prevx, prevy)를 openGl 좌표계로 변환해야 한다.

그 함수는 아래와 같다.

void convert\_window\_coord\_to\_openGL\_coord(int x, int y, float\* converted\_x, float\* converted\_y) {

float ex = (float)window\_width / 2;

float ey = (float)window\_height / 2;

\*converted\_x = (x - ex) / 250;

\*converted\_y = (-y + ey) / 250;

}

위의 정보들을 통해, (px, py)를 현재 인자로 넘어온 (x,y)의 위치로 변환가능하다.(picking)

그 Translation 코드(mousemove()의 일부)는 아래와 같다.

if (active\_key == GLUT\_ACTIVE\_SHIFT && leftbuttonpressed) {

if ((converted\_prevx <= px + 0.02 && px - 0.02 <= converted\_prevx)

&& (converted\_prevy <= py + 0.02 && py - 0.02 <= converted\_prevy)) {

px = converted\_x;

py = converted\_y;

//계속 누르고 있는 상태이므로 클릭을 시작한 위치를 갱신해야 한다.

prevx = x;

prevy = y;

glutPostRedisplay();

}

}

클릭의 오차범위는 openGl 좌표계로 0.02 x 0.02 로 하였다.

(f) 다각형 이동

(prevx, prevy)는 마우스를 클릭한 시작한 위치, 이전의 마우스 커서 위치를 나타낸다고 했다.

delx = x - prevx, dely = y - prevy 만큼 다각형을 이동하면 된다.

나머지는 (e)에서 설명한 정보들로 Translation 구현이 가능하다.

그 코드(mousemove()의 일부)는 아래와 같다.

if (active\_key == GLUT\_ACTIVE\_ALT && rightbuttonpressed) {

float delx = converted\_x - converted\_prevx;

float dely = converted\_y - converted\_prevy;

object[0][0] += delx;

object[0][1] += dely;

object[1][0] += delx;

object[1][1] += dely;

object[2][0] += delx;

object[2][1] += dely;

object[3][0] += delx;

object[3][1] += dely;

object[4][0] += delx;

object[4][1] += dely;

object[5][0] += delx;

object[5][1] += dely;

object\_center\_x = object\_center\_y = 0.0f;

for (int i = 0; i < n\_object\_points; i++) {

object\_center\_x += object[i][0];

object\_center\_y += object[i][1];

}

object\_center\_x /= n\_object\_points;

object\_center\_y /= n\_object\_points;

//계속 누르고 있는 상태이므로 클릭을 시작한 위치를 갱신해야 한다.

prevx = x;

prevy = y;

glutPostRedisplay();

}

(e)와 달리 그냥 alt, 오른쪽 마우스만 활성화 되있다면 이동시키는 것을 볼 수 있다.

(g) 다각형 축소 확대

원점으로 이동한 후(T(-Gx, -Gy)), 물체의 크기를 isotropic 하게 변환하고(S(s, s)), 다시 원위치 시킨다(T(Gx, Gy)).

이와 같은 아핀 변환을 이용해 Scaling 변환 구현이 가능하다.

그리고 마우스가 왼쪽으로 이동할 경우 다각형이 작아져야 하므로 s는 1보다 작은 양수여야 한다.

그리고 마우스가 오른쪽으로 이동할 경우 다각형이 커져야 하므로 s는 1보다 큰 양수여야 한다.

그 코드(mousemove()의 일부분)는 아래와 같다.

if (active\_key == GLUT\_ACTIVE\_CTRL && rightbuttonpressed) {

float delx = converted\_x - converted\_prevx;

//delx가 음수인 경우 1보다 작은수 양수를, 양수인 경우 1보다 큰 양수를 얻어야한다.

float s = delx + 1;

while (s < 0) {

++s;

}

object[0][0] = s \* (object[0][0] - object\_center\_x) + object\_center\_x;

object[0][1] = s \* (object[0][1] - object\_center\_y) + object\_center\_y;

object[1][0] = s \* (object[1][0] - object\_center\_x) + object\_center\_x;

object[1][1] = s \* (object[1][1] - object\_center\_y) + object\_center\_y;

object[2][0] = s \* (object[2][0] - object\_center\_x) + object\_center\_x;

object[2][1] = s \* (object[2][1] - object\_center\_y) + object\_center\_y;

object[3][0] = s \* (object[3][0] - object\_center\_x) + object\_center\_x;

object[3][1] = s \* (object[3][1] - object\_center\_y) + object\_center\_y;

object[4][0] = s \* (object[4][0] - object\_center\_x) + object\_center\_x;

object[4][1] = s \* (object[4][1] - object\_center\_y) + object\_center\_y;

object[5][0] = s \* (object[5][0] - object\_center\_x) + object\_center\_x;

object[5][1] = s \* (object[5][1] - object\_center\_y) + object\_center\_y;

//계속 누르고 있는 상태이므로 클릭을 시작한 위치를 갱신해야 한다.

prevx = x;

prevy = y;

glutPostRedisplay();

}

(h) 추가 구현

ctrl 키를 누르고 왼쪽 마우스를 클릭하면 다각형을 y=x 축을 기준으로 대칭이동을, alt 키를 누르고 왼쪽 마우스를 클릭하면 선분을 y=x 축을 기준으로 대칭이동 시키는 기능을 구현하였다.

y=x축을 x축으로 이동 시킨 후(R(-45)), x축에 대해 대칭이동 시키고(R\*), 다시 y=x를 원위치로 이동시킨다(R(45)).

이 식을 정리하면, 결국 (x,y) -> (y,x) 로 바꾸는 것과 같다.

이를 구현한 코드(mousepress() 함수의 일부분)는 아래와 같다.

case GLUT\_LEFT\_BUTTON: {

leftbuttonpressed = 1;

if (active\_key == GLUT\_ACTIVE\_CTRL) {

for (int i = 0; i < 6; ++i) {

float temp = object[i][0];

object[i][0] = object[i][1];

object[i][1] = temp;

}

object\_center\_x = object\_center\_y = 0.0f;

for (int i = 0; i < n\_object\_points; i++) {

object\_center\_x += object[i][0];

object\_center\_y += object[i][1];

}

object\_center\_x /= n\_object\_points;

object\_center\_y /= n\_object\_points;

glutPostRedisplay();

}

if (active\_key == GLUT\_ACTIVE\_ALT) {

float temp = px;

px = py;

py = temp;

temp = qx;

qx = qy;

qy = temp;

glutPostRedisplay();

}

break;

}