**기초 컴퓨터 그래픽스**

**HW1 README**

20211584 장준영

1. 환경 명세

Windows 10 64bit, AMD Ryzen 5 5600X, RTX 3070Ti, Visual Studio 2022 Release x64

2. 요구사항

(a) 윈도우 화면

- 구현 여부 : 예

- 작동 확인 방법 : 프로그램 실행

- 구현 방법 : main 함수에서 실행될 윈도우의 사이즈를 설정할 수 있다. 제출된 소스 파일의 414~416 line은 윈도우를 초기화해주는데, glutInitWindowSize(750, 750);를 통해 윈도우의 사이즈를 가로, 세로 각각 750픽셀로 설정하였다.

(b) 선분 그리기

- 구현 여부 : 예

- 작동 확인 방법 : 프로그램 실행

- 구현 방법 : initialize\_renderer 함수에서 선분의 속성을 초기화하고 draw\_line 함수를 호출해 선분을 윈도우에 그렸다. 전역 변수인 px, py, qx, qy가 선분의 시작점과 끝점을 나타내는데, 이를 initialize\_renderer 함수에서 값을 지정하고, draw\_line 함수에서 gl~ 함수를 통해 윈도우에 그렸다.

(c) 비대칭 다각형 그리기

- 구현 여부 : 예

- 작동 확인 방법 : 프로그램 실행

- 구현 방법 : initialize\_renderer 함수에서 다각형의 속성을 초기화하고 draw\_object 함수를 호출해 선분을 윈도우에 그렸다. 전역 변수인 object[6][2]와 object\_center가 각각 다각형의 6개의 점과 기준점이 될 무게중심을 나타내는데, 이를 initialize\_renderer 함수에서 값을 지정하고, draw\_object 함수에서 gl~ 함수를 통해 윈도우에 그렸다.

(d) Rotation 변환 기능

- 구현 여부 : 예

- 작동 확인 방법 : 휠을 위로 돌리면 반시계 방향으로, 아래로 돌리면 시계 방향으로 회전

- 구현 방법 : 휠의 동작을 이용하기 위해 glutMouseWheelFunc(wheelmove)를 callback에 등록하고 wheelmove(int wheel, int direction, int x, int y) 함수를 정의했다. wheelmove에선 휠을 위로 돌리면 direction이 음수가 되고, 아래로 돌리면 양수가 된다. 각각의 동작에 따라 하얀 점이 회전할 수 있도록 affine transformation의 행렬을 계산했다.

회전 기준점, 즉 파란 점의 좌표가 (X,Y)이면, 선분의 축이 원점이 되도록 이동시키고, 원하는 만큼 회전한 후, 다시 원점으로 되돌려야 한다. T(X,Y)R(theta)T(-X,-Y) 행렬을 계산하여 이동 후 좌표인 x’, y’를 구하면 위와 같다. 다음 식을 285~295 line의 wheelmove 함수에 그대로 적용하였다.

(e) Picking 기능

- 구현 여부 : 예

- 작동 확인 방법 : Shift와 마우스 왼쪽 버튼을 동시에 누르고 선분의 파란 점을 클릭하여 이동 동작 확인

- 구현 방법 : 드래그를 통해 물체의 위치를 변환할 땐 glutMouseFunc(mousepress)와 glutMotionFunc(mousemove)를 이용했다. mousepress에선 기본적으로 마우스의 양쪽 버튼이 눌린 상태와 눌리지 않은 상태, 즉 네 가지 경우에 대해 인식할 수 있도록 구현하였다. tempx와 tempy를 통해 설정한 좌표계의 좌표로 변환한 마우스의 위치를 저장하였다. 버튼이 눌렸을 땐 눌린 상태를 알려주는 rightbuttonpressed, leftbuttonpressed의 값을 1로 바꾸며 prevx와 prevy의 값을 클릭 위치로 저장하였고, 눌리지 않았을 땐 -pressed 변수의 값을 0으로 바꾸었다. 특히, Picking 기능을 위해 선분의 파란 정점 위에서 왼쪽 버튼을 눌렀을 땐 전역 변수로 선언한 pressedOnVertex의 값을 1로 바꾸고, 왼쪽 버튼을 들었을 땐 0으로 바꾸었다. 마우스가 정점과 완벽하게 같은 좌표를 짚기는 어렵기 때문에, 0.02f의 오차는 허용하였다.

이후 마우스가 이동하는 부분에선 mousemove를 이용한다. 마우스가 얼마나 이동하였는지 알기 위해 변화량인 dx, dy를 각각 이전 좌표와 현재 좌표의 차이인 x-prevx, prevy-y로 설정하였다.(y축이 윈도우 좌표계와 설정한 좌표계와 방향이 다르기 때문에 부호가 반대이다.) 이후 이 변화량을 설정한 좌표계의 기준으로 tempx와 tempy에 저장하였다. Picking을 할 땐 pressedOnVertex가 1이고 Shift가 동시에 눌려있어야 하므로, 264~268 line의 조건문과 glutGetModifiers() 함수를 사용하였다. 모든 조건을 만족하는 경우에 현재 파란 정점의 위치를 전역적으로 저장한 px와 py의 값을 tempx와 tempy만큼 이동시켜 구현하였다.

(f) Translation 변환 기능

- 구현 여부 : 예

- 작동 확인 방법 : Alt와 마우스 오른쪽 버튼을 동시에 누르고 마우스를 움직이며 동작 확인

- 구현 방법 : (e)에서 구현한 picking과 대부분 동일하지만, 마우스를 클릭한 지점에 대한 조건이 없다. 따라서 251~257 line에 rightbuttonpressed가 1이고, specialKey=glutGetModifiers()가 GLUT\_ACTIVE\_ALT인 경우 동작이 실행되도록 조건문을 작성했다. 다각형을 이루는 모든 정점인 object[6][2]와 object\_center\_x, object\_center\_y가 모두 tempx, tempy만큼 변할 수 있도록 조건문을 사용하였다.

(g) Scaling 변환 기능

- 구현 여부 : 예

- 작동 확인 방법 : Ctrl와 마우스 오른쪽 버튼을 동시에 누르고 x축의 음, 양의 방향으로 움직이며 동작 확인

- 구현 방법 : mouse와 specialKey를 이용하기 때문에 동작 조건 설정은 (e), (f)와 동일하다. 258~263 line에 specialKey가 GLUT\_ACTIVE\_CTRL이고 오른쪽 버튼이 눌린 상태에 동작하도록 조건문을 설정하였다. Scaling 변환은 원점을 기준으로 하는 변환이므로, 회전과 마찬가지로 기준점을 원점으로 옮겨 Scaling을 진행한 후 다시 원위치로 돌리는 작업을 해야 한다.

파란 정점, 즉 기준점의 좌표가 (X,Y)인 경우 T(X,Y)S(a,b)T(-X,-Y)의 행렬을 곱해야 한다. 변환 이후의 점 x’, y’의 좌표는 위와 같다. 260~261 line에 이를 동일하게 구현하였다. x축의 음의 방향으로 마우스를 이동시키면 tempx가 음수이고 반대의 경우 양수이므로, 어떤 수를 tempx만큼 제곱한 값을 확장/축소 비율인 a=b로 사용하면 확장의 경우 1보다 크게, 축소의 경우 작게 마우스의 이동 범위만큼 Scaling할 수 있다. object[6][2]의 모든 점이 변환되도록 반복문을 사용하였지만, scaling은 무게중심을 변화시키지 않으므로 object\_center\_-는 그대로 두었다.

(h) 추가 다각형과 기하 변환

- 구현 여부 : 예

- 작동 확인 방법 : 윈도우가 처음 떴을 때 보이는 왼쪽 하단의 정사각형이 추가 다각형임. 다각형의 무게중심인 파란 정점을 마우스 왼쪽 버튼과 오른쪽 버튼으로 누르며 Shearing과 Scaling, 배경색 변화 확인(구현 방법 항목의 강조된 부분에 자세히 설명되어 있음)

- 구현 방법 : 추가 다각형(object2)은 위에서 구현한 비대칭 다각형(object)과 동일하게 object2[4][2], object\_center\_x, object\_center\_y, n\_object2\_points, object2prev[4][2], xlen, ylen, centx, centy를 전역 변수로 갖고있다. 마찬가지로 initialize\_renderer에서 xlen, ylen, centx, centy를 사용해 위치와 크기를 초기화하고, draw\_object2(void)로 윈도우에 그려냈다. object2prev는 object2가 scaling 변환되기 전의 좌표를 저장하는 용도이다. object2는 두 가지 동작을 한다.

1) 파란 정점을 마우스 왼쪽 버튼으로 누르면 x축 방향 shearing을 하고, 오른쪽 버튼을 누르면 y축 방향 shearing을 한다.

2) 마우스 왼쪽 버튼을 한 번 누를 때마다 배경색의 r값이 0.01f만큼 증가하고, 오른쪽 버튼을 한 번 누를 때마다 g값이 0.01f만큼 증가한다. 클릭을 15번 할 때마다 object2가 다시 정사각형이 되며 확장 Scaling이 진행되고, 배경색의 b값이 0.1f만큼 증가하고 r, g값은 다시 되돌아간다.

1)의 기능을 구현하기 위해 (e)의 Picking과 비슷한 방법을 사용하였다. mousepress 함수에서 파란 정점의 좌표인 object2\_center\_-에서 마우스 버튼이 눌렸을 때 Shearing 연산과 배경색 변환을 하도록 하였다. 184~190 line과 199~205 line에 작성했다.

Shearing 역시 scaling, rotation과 같이 원점으로 기준을 이동한 후 진행하여 원위치로 되돌려야 한다. T(X,Y)Sh(\*)T(-X,-Y) 행렬을 곱해 나온 좌표를 그대로 적용하였다.

2)에서 마우스가 몇 번 눌렸는지 세기 위해 static int clickcnt를 사용한다. 211~239 line에서 clickcnt%15==14가 될 때마다 동작하는 조건문을 작성하였다. shearing이 진행되기 이전의 정사각형이 object2prev에 저장되어 있으므로 다시 object2에 할당하고, 1.5f만큼 scaling을 진행한다. 이후 커진 정사각형의 좌표를 다시 object2prev에 저장한다. 배경색의 r, g값은 다시 되돌리고 b값은 0.1f만큼 증가시킨다. b값이 0.75f 이상이 되면 b값도 초기값으로 다시 되돌린다.