**기초 컴퓨터 그래픽스**

**HW1 README**

학번 20171273 / 이름 심현우

**1. [환경 명세]**

window10 64bit, i7-9750H, GTX 1660 Ti, visual studio 2022 – win 64, release

**2. [요구사항]**

(a) 윈도우 화면

- 구현 여부: 예

- 작동 확인 방법: 실행하면 바로 뜸

- 구현 방법: main블록에서 window에 대한 사이즈와 초기 위치를 glutwindowsize와 glutwindowposition으로 설정해주고 glutcreatewindow함수를 통해 window창을 띄운다. 본인은 windowsize를 제공된 HW1 예제 실행파일과 같이 800X600으로 설정하고 window의 초기 위치는 550, 200으로 설정했다. 또한, 초기 윈도우 화면의 배경 색상은 display함수에서 설정해줬다. display함수는 윈도우에 출력을 담당하는 함수로 콜백을 통해 호출된다. 이 콜백 또한 main함수에서 호출되어 초기 상태를 윈도우 화면에 출력할 수 있다. 배경 또한 실행 예시 파일과 같은 연두색인 80, 200, 120로 설정했다.

(b) 점 그리기

- 구현 여부: 예

- 작동 확인 방법: shift키를 누르고 왼쪽 마우스 버튼을 클릭한다.

- 기타 요구사항에 명시된 내용: shift 키를 누르지 않은 상태에서는 작동하지 않는다

- 구현 방법: 위에 잠시 설명했던 것처럼 윈도우 화면의 출력은 콜백을 통해 이루어진다. 이 콜백에는 display뿐만 아니라 glutkeyboardfunc, glutspecialfunc, glutmousefunc, glutmotionfun, glutreshapefun, glutclosefunc도 포함되어 있다. Display는 화면에 출력하는 것을 담당한다. 따라서 어떤 변화에 대한 출력은 display함수 안에 설정해야 한다. 점 그리기는 shift를 누른 상태에서 마우스 좌클릭을 통해 출력해야 한다. Mouse 클릭에 대한 입력은 glutmousefunc에서 다룬다. glutmousefunc에는 사용자가 작성한 함수를 파라미터로 받아 사용하는데 이때 함수는 4가지의 int 파라미터가 들어간다. 본인은 mousepress라는 함수를 설정하여 glutmousefunc의 파라미터로 넣어줬다. Mousepress의 파라미터는 button, state, x, y로 각각 button은 마우스 클릭에 대한 정보, state는 클릭의 상태에 대한 변화, 즉 on->off인지, off->on인지에 대한 정보, x와 y는 클릭 당시의 마우스의 윈도우에서 좌표를 의미한다. 이때 좌표는 int로 좌상단 0,0에서 시작하여 우하단 800,600이 된다. (b)의 요구사항은 shift를 누른 상태에서 좌클릭 시 점을 찍는 것이다. Shift를 누른 상태는 glmodifier함수를 통해 설정할 수 있다. If문을 통해 shift가 활성화된 상태인지 확인하고, button==GLUT\_LEFT\_BUTTON을 통해 좌클릭이 이루어졌는지, state가 클릭 당시인지를 확인한다. Shift와 클릭이 활성화 되면 그 위치를 저장해야 한다. 점은 최대 128개까지 찍을 수 있으므로 전역변수인 MAX\_POINT 128과 float point[MAX\_POINT][2]를 선언했다. 점의 위치는 x와 y를 통해 저장하는데 display에서 glvertex2f를 통해서 점을 그리기 위해 float로 위치를 저장했다. 현재 window사이즈인 current\_width와 current\_height를 사용하여 점의 x좌표는 ((x / current\_width)-0.5)\*2로, y좌표는 상하의 부호가 반대이므로 (((current\_height-y) / current\_height)-0.5)\*2로 설정했다. 이를 point배열에 저장하고 point\_count를 통해 총 점의 개수와 그 점의 위치를 각각 계속 저장한다. 이를 display함수 내부에서 for문을 통해 저장한 모든 점에 대한 위치를 glbegin과 glpointsize, glcolor, glvertex2f, glend 함수를 통해 출력한다. 점은 파란색 0, 0, 255으로 설정했고 크기는 5.0으로 했다.

(c) 선 그리기

- 구현 여부 : 예

- 작동 확인 방법 : shift키를 누른 상태에서 마우스 좌클릭을 2번 이상 해본다.

- 기타 요구 사항 : shift키를 누르지 않은 상태에선 작동하지 않는다.

- 구현 방법 : 위의 점 그리기와 매우 유사하다. 우선 점의 정보가 저장되어 있는 상태에서 display의 선을 그리기 위한 함수만 추가한다. Gllinewidth, glbegin, glcolor3f, glend, glvertex2f 두개를 사용하여 선의 양 끝점의 정보를 넣는다. 선 또한 for문을 통해 처음 점부터 마지막 점까지 사이를 선으로 연결한다. 선은 빨간색 128, 0, 0으로 설정했고 굵기는 1.0이다.

(d) P키로 polygon완성

- 구현 여부 : 예

- 작동 확인 방법 : 점을 3개 이상 만들고 p키를 누른다.

- 구현 방법 : b에서와 같이 점의 정보를 저장하는 방법으로 polygon을 완성한다. 여기서는 p키에 대한 입력을 받아야 하므로 glutkeyboardfunc을 사용한다. 이 함수의 파라미터로 직접 작성한 keyboard가 들어간다. keyboard함수는 key와 x, y를 파라미터로 사용하는데 key는 누르는 키에 대한 정보, x, y는 키를 눌렀을 때 마우스의 위치에 대한 정보이다. Keyboard 함수 내부에 if문을 통해 key==’p’인 경우를 추가하고 이 때, polygon을 완성하도록 내부를 작성한다. Polygon은 마지막 점과 초기 점을 선으로 연결하면 되므로 마지막 점의 다음 점을 초기점과 같은 정보로 저장하면 된다. 그렇게 설정하면 display함수를 변경하지 않아도 해당 요구사항을 만족할 수 있다. 따라서 point\_count에 초기 점의 정보를 저장하고 point\_count를 하나 증가시킨다.

(e) C로 점, 선 초기화

- 구현 여부 : 예

- 작동 확인 방법 : 점을 하나 이상 찍고 c키를 누른다.

- 구현 방법 : 지우는 방법은 점의 정보를 제거함으로써 만족시켰다. 결국 점과 선의 출력은 점의 정보를 통해 이루어지므로 c키를 누르면 point array를 초기화 시키는 방법으로 구성했다. d처럼 keyboard함수에서 c키 입력에 대한 함수를 구성했다. if문을 통해 c키가 눌린 것을 확인하면, for문을 통해 모든 점의 정보를 NULL로 초기화한다.

(f) 3개 미만 점일 때 p키 에러 메시지

- 구현 여부 : 예

- 작동 확인 방법 : 3개 미만 점을 찍고 p키를 누른다.

- 구현 방법 : polygon을 완성하기 위해서는 3개 이상 점이 필요하므로 3개 미만 점을 찍고 p를 눌렀을 때 에러메세지를 출력한다. 이는 p키 입력 내부에서 처리해줬는데 if문을 통해 point\_count가 3보다 작으면 예시 실행 파일과 같이 choose at least three points를 fprintf를 통해 터미널에 출력하고 윈도우 화면에는 출력의 변화가 없도록 했다.

(g) polygon 시 shift 좌클릭 반응 X

- 구현 여부 : 예

- 작동 확인 방법 : 3개 이상 점을 찍고 p키를 누른 상태에서 shift를 누른 상태에서 좌클릭을 한다.

- 구현 방법 : polygon을 완성한 상태에서는 다른 점을 찍지 않도록 한다. 이는 p\_flag를 통해서 구현했다. P\_flag는 전역변수로 선언하고 초기값은 0으로 할당했다. keyboard내부 함수에서 point\_count가 3이상인 if문, 즉 polygon이 완성되는 코드에서 마지막에 p\_flag=1로 설정했다. 이는 p키가 눌려 다각형이 완성되었을 때, p\_flag는 1의 상태를 갖는다는 뜻이다. 이 후, mousepress 내부의 shift와 button의 if문 조건에 p\_flag==0을 추가하여 다각형이 완성된 상태에는 클릭에 반응하지 않도록 한다. P\_flag는 c키가 눌렸을 때, 즉 다각형이 초기화 되었을 때 다시 0으로 설정하도록 한다.

(h) polygon 시 방향키로 polygon 움직이기

- 구현 여부 : 예

- 작동 확인 방법 : 다각형을 완성하고 상하좌우 방향키를 누른다.

- 기타 요구 사항 : 다각형이 완성된 상태

- 구현 방법 : keyboard는 특수키를 제외한 나머지 키에 대한 입력에 대해 다룬다. 특수키 즉, shift, insert, ctrl, 방향키 등은 special에서 따로 다룬다. 이를 나타내는 콜백함수는 glutspecialfunc이다. 이 함수의 파라미터로 본인이 작성한 special함수가 들어간다. special함수의 파라미터는 keyboard와 같이 key, x, y이다. Key는 특수키를 의미하고 x, y는 특수키가 눌렸을 때의 마우스 커서의 위치 정보이다. h에서는 특수키인 방향키에 대한 입력을 다뤄야하므로 switch구문을 통해 작성했다. Key에 각각 GLUT\_KEY\_LEFT … GLUT\_KEY\_UP을 switch구문을 통해 구분하고 각각 키가 눌렸을 때, point array의 점 위치 정보에 일정한 값을 가감하므로써 다각형이 움직일 수 있도록 한다. 예를 들어, GLUT\_KEY\_LEFT에서는 왼쪽 방향키를 눌렀을 때이므로, point array의 각 x좌표에 -0.05를 해주어 왼쪽으로 0.05만큼 움직일 수 있도록 한다. 나머지 방향키도 같은 방법으로 작성했다. 기타 요구 사항인 다각형이 완성된 상태에서만 움직일 수 있도록 switch 구문의 키에 대한 입력을 받을 때 상위 구문으로 if를 사용하여 p\_flag==1, 즉 다각형이 완성된 상태에서만 구문에 들어갈 수 있도록 제한한다.

(i) polygon 시 마우스 우클릭 드래그로 polygon 움직이기

- 구현 여부 : 예

- 작동 확인 방법 : 다각형을 완성한 상태에서 마우스 우클릭을 유지하며 마우스 커서 이동

- 기타 요구 사항 : 다각형이 완성된 상태

- 구현 방법 : 위의 h와 유사하다. 다각형이 완성된 상태에서 이번에는 특수키가 아닌 마우스 클릭을 입력으로 받는다. 마우스 우클릭 유지에 대한 입력은 mousepress에서 작성했다. 우선 위의 좌클릭과는 반대로 GLUT\_RIGHT\_BUTTON을 사용하여 우클릭에 대한 button입력을 받고, 클릭에 대한 유지를 원하기 때문에 state의 GLUT\_DOWN과 GLUT\_UP으로 코드를 작성했다. State가 GLUT\_DOWN이면 클릭이 유지되고 있는 상황으로 이때의 rightbuttonpressed값을 1로 할당해주어 출력을 나타내도록 한다. 이때의 출력은 mousemove에서 다룬다. Mousemove는 마우스 클릭 유지 시 움직임에 대한 함수인데, 이는 콜백에서 glutmotionfunc의 파라미터로 사용된다. 본인이 작성한 mousemove는 x, y를 파라미터로 받는데 이는 마우스 커서의 위치에 대한 정보이다. mousemove에서 위의 rightbuttionpressed의 값을 1로 받으면 이는 우클릭이 유지되었다고 인지하고 마우스 움직임에 따라 다각형을 움직일 수 있도록 한다. 이때 변화된 마우스의 움직임을 x, y, 이전 점의 정보를 prev\_x, prev\_y로 저장하여 변화된 정도에 대한 값을 계산하고 이를 점의 정보에 가감하여 새로운 다각형의 위치를 나타낸다. 위와 마찬가지로 다각형이 완성된 상태에서만 움직일 수 있도록 mousemove의 rightbuttonpressed==1의 if 구문에 p\_flag==1을 추가한다.

(j) polygon 시 r키로 회전

- 구현 여부 : 예

- 작동 확인 방법 : polygon을 완성한 상태에서 r키를 누른다.

- 기타 요구 사항 : 다각형이 완성된 상태

- 구현 방법 : 우선 무게 중심을 중심으로 회전하기 때문에 무게중심부터 구해준다. 이는 keyboard의 if 구문에서 key==’r’, 즉 r키가 눌렀을 경우 내부에 작성했다. 무게 중심을 구하는 공식은 다각형을 삼각형으로 쪼개서 무게 중심을 이은 다각형의 무게 중심을 다시 구하는 방법을 사용했다. 구한 무게 중심은 전역 변수로 선언한 center[2]에 저장했다. 이후 center\_flag를 전역변수로 선언하고 초기값을 0으로 할당했다. r키가 눌렀을 경우 center\_flag=1로 바꾸고 이를 display에 if구문에 묶어 무게중심을 구하면 이를 화면에 출력할 수 있도록 했다. 무게 중심을 기준으로 회전을 나타내기 위해 rotate\_flag변수도 선언했다. Rotate\_flag는 현재 다각형이 회전하고 있는지에 대한 변수이다. 이것이 0이면 회전하지 않는 것이고 1이면 회전하는 것이다. 초기값은 0으로 r키가 눌렀을 경우 rotate\_flag==0인 것을 확인하면 무게중심을 구하고 rotate\_flag를 1로 바꾼다. 이후 display에서 rotate\_flag가 1인 것을 확인하면 회전을 진행한다. 지속적인 회전을 나타내기 위해 timer를 설정했다. timer함수를 본인이 선언하여 내부에 gluttimerfunc를 통해 일정 시간마다 함수가 불릴 수 있도록 했다. 이를 main함수에 선언하여 프로그램을 실행하고 나면 display가 일정 시간마다 새로 출력된다. 이를 이용하여 display에서 rotate\_flag==1을 감지하고 이 내부에 회전을 위한 계산을 진행한다. 회전은 translate->rotate->translate를 통해 구성했다. 우선 무게중심을 기준으로 회전하기 위해 무게중심을 0,0 즉 기준점으로 잡아줘야 하고 이때, translate를 한다. 이는 각각 점 위치정보에 center정보를 빼는 것으로 할 수 있다. 이후 rotate를 위해 math.h라이브러리의 cos과 sin함수를 불러와 계산하고, 변환하는 각은 0.03f미리 전역 변수로 선언했다. Rotate를 진행하면 다시 원래의 자리로 돌아가야 하기 때문에 무게중심을 빼 준만큼 다시 더해줬다. 마지막 기타 요구 사항으로 다각형이 완성된 상태에서 작동해야 하므로 keyboard의 r키에 대한 입력 상위 구문에 if, p\_flag==1을 작성한다.

(k) 회전 중 무게중심 출력

- 구현 여부 : 예

- 작동 확인 방법 : 다각형을 완성하고 r키를 누른다.

- 구현 방법 : 위의 설명한 무게 중심을 구하고 center\_flag를 통해 무게중심을 화면에 출력한다.

(l) 회전 중 r키 입력 시 멈추기

- 구현 여부 : 예

- 작동 확인 방법 : 다각형이 완성되고 r키를 누른 후 다각형이 회전하는 상태에서 r키를 한번 더 누른다.

- 기타 요구 사항 : 회전이 멈추고 무게중심이 사라진다.

- 구현 방법 : r키를 누르면 위에 설명했던 것처럼 초기 rotate\_flag값이 1로 바뀌게 된다. 이는 현재 다각형이 회전하고 있는 상태를 의미한다. 이미 회전하고 있는 상태에서 r키를 다시 누르는 상황을 keyboard의 key==’r’에 작성한다. 이때, 위의 상황과 달리 if(rotate\_flag==1)을 감지하여 작성한다. 내부 코드는 우선 무게중심을 지우기 위해 center[2]의 값을 null로 초기화한다. 또한 rotate\_flag와 center\_flag를 0으로 초기화하여 회전 출력과, 무게중심 출력을 모두 제거한다.

(m) 회전 중 그리기, 지우기, 움직이기 불가

- 구현 여부 : 예

- 작동 확인 방법 : r키를 누르고 다각형이 회전하는 상황에서 다른 키들을 눌러본다.

- 기타 요구 사항 : 회전하고 있는 상황에서는 키를 사용할 수 없게 한다.

- 구현 방법 : mousepress, mousemove, keyboard, special 등 커맨드에 대한 동작을 나타내는 구문에서 상위 if에 rotate==0을 추가하여 rotate==1인 상황, 즉 회전하는 상황에서는 동작할 수 없도록 제한한다.

(n) 윈도우 크기 변경 시 비율 유지

- 구현 여부 : 예

- 작동 확인 방법 : 윈도우 화면을 늘리고 줄여본다.

- 기타 요구 사항 : 비율을 유지한다.

- 구현 방법 : 결과적으로 다각형을 그리는 요소는 점의 정보이다. 이를 출력할 때 float를 사용하여 윈도우의 크기에 따라 비율을 유지하도록 출력하는 상태이다. 때문에 윈도우의 크기를 조정하는 함수에 대해서 구성할 필요가 있다. 우선 전역 변수에 initial width, height와 current width, height를 선언했다. 각각 변수 의미 그대로 처음 윈도우 크기, 현재 윈도우 크기를 의미한다. 첫 선언은 모두 width=800, height=600으로 설정했다. 이후 변화하는 윈도우의 크기는 current 변수를 통해 진행한다. 윈도우 크기를 조정하는 콜백 함수는 glutreshapefunc이다. 이 함수의 파라미터로 함수가 들어오는데 이를 본인이 reshape로 선언했다. reshape함수의 파라미터로 width와 height가 들어온다. 이는 현재 변화된 윈도우의 크기를 의미한다. 따라서 이 함수 내부에 current\_width, height를 파라미터인 현재 윈도우 크기로 설정했다. 비율 유지에 대한 점의 정보는 미리 current\_width, height를 사용했기 때문에 문제 없다.

(o) f키로 종료

- 구현 여부 : 예

- 작동 확인 방법 : f키를 누른다.

- 구현 방법 : f키를 누르는 입력은 keyboard 함수에서 다룬다. Key==’f’케이스에서 glutleavemainloop을 호출하여 main함수를 빠져나오는, 즉 프로그램을 종료하는 동작을 수행한다.

(p) 모든 점, 선, 무게중심 다른 색 사용

- 구현 여부: 예

- 작동 확인 방법: 위의 요구사항들 실행 중 확인 가능

- 윈도우 배경은 연두색, 점은 파란색, 선분은 빨간색, 무게 중심 점은 노란색으로 그려지게 설정하였다. 이는 예시 실행 파일을 보고 설정하였다.