**기초 컴퓨터 그래픽스**

**HW1 README**

20212020 박민준

**1. [환경 명세]**

OS: Windows 11 Home 64bit

## CPU: Intel(R) Core(TM) i5-1240P @ 1.70GHz

## RAM: 16.00GB

## Compiler: Visual Studio 2022 Release Mode/x64 Platform

**2. [요구사항]**

- 모든 요구사항을 구현하였다.

**[Definition.h]**

- SELECTED\_LINE\_COLOR 매크로를 자홍색(1.0f, 0.0f, 1.0f)으로 추가하였다.

- Status 구조체에 leftbuttonpressed, selected 변수를 추가하였다. leftbuttonpressed는 왼쪽 마우스 클릭 여부를 저장하기 위한 변수이고, selected는 무게중심 클릭 시 색상 변경을 위한 변수이다.

- MAX\_POSITIONS 값을 128로 수정하여 사용자가 선택하는 점을 최대 128개로 제한하였다.

- 다른 파일에서 Window, Status, My\_Polygon 구조체를 사용할 수 있도록 전역 변수로 선언하였다.

**[Polygon.cpp]**

- draw\_lines\_by\_points() 함수에서 무게중심 클릭 시 선 색깔이 변경되도록 구현하였다.

- 무게중심도 같이 이동시키는 함수인 move\_points\_with\_center() 함수와 무게중심을 기준으로 다각형을 확대/축소시키는 zoom\_polygon() 함수를 추가하였다.

**(a) 다각형 생성 모드**

- 구현 여부: 예

- 작동 확인 방법:

1) 프로그램 실행 후 p키를 누르면 다각형 생성 모드로 진입한다.

2) Ctrl 키를 누른 상태에서 오른쪽 마우스 클릭 시 점이 추가된다.

3) 두 점까지는 선분으로, 세 점부터는 다각형 형태로 연결된다.

4) p키를 다시 누르면 다각형이 완성된다.

5) c키를 누르면 다각형이 삭제된다.

- 구현 방법:

- mouse\_press() 함수 내에서 다각형 미완성 상태(st.polygon\_mode == 0)일 때, Ctrl + 마우스 오른쪽 버튼을 클릭하면 add\_point() 함수를 호출하여 점을 추가한다.

- draw\_lines\_by\_points() 함수에서 점이 2개 이상일 때 GL\_LINE\_LOOP를 이용해 선택된 순서대로 선을 그린다.

- keyboard() 함수에서 p키 입력 시 점 개수가 3개 이상일 경우, polygon\_mode 상태를1로 설정하고 update\_center\_of\_gravity() 함수를 호출하여 무게 중심점에 해당하는 pg.center\_x, pg.center\_y 값을 업데이트한다.

**(b) 다각형 무게 중심 표시**

- 구현 여부: 예

- 작동 확인 방법: p키를 두 번째로 눌러 다각형이 완성되면 중심에 빨간 점이 자동으로 표시된다.

- 구현 방법:

- update\_center\_of\_gravity() 함수에서 다각형의 각 점 좌표를 평균 내어 중심 좌표(pg.center\_x, pg.center\_y)를 계산한다.

- display() 함수에서 GL\_POINTS를 사용하여 다각형 완성 후 중심 좌표를 빨간 점으로 시각화 한다.

**(c) 이동 모드 (중심 클릭 시 선택 & 색 변경)**

- 구현 여부: 예

- 작동 확인 방법: 다각형 완성 후, 중심점을 마우스 왼쪽 클릭하면 선 색이 자홍색으로 바뀐다.

- 구현 방법:

- mousepress() 함수에 클릭 판정 로직을 추가한다. GLUT\_LEFT\_BUTTON 클릭 시, 현재 마우스 위치를 정규화 좌표로 변환하고 중심점 주변 반경(점 크기 기준)과 비교한다. 만약 클릭되었다면 st.leftbuttonpressed 변수와 st.selected 변수를 1로 설정한다.

- draw\_lines\_by\_points() 함수에서 st.selected == 1일 경우 선의 색상을SELECTED\_LINE\_COLOR로 변경하여 그린다.

**(d) 마우스 드래그 이동**

- 구현 여부: 예

- 작동 확인 방법:

1) 중심점 클릭 후 마우스를 드래그 하면, 다각형과 중심점이 함께 이동한다.

2) 이동 중에도 선 색은 계속 자홍색을 유지한다.

- 구현 방법:

- mousemove() 함수 내에 이동 처리 로직을 추가한다. st.leftbuttonpressed == 1 조건일 때 이전 마우스 좌표와 현재 좌표 차이를 계산한다.

- 이동 벡터를 정규화 좌표계 기준으로 환산한 후 move\_points\_with\_center() 함수를 호출하여 모든 점과 중심 좌표를 함께 이동시킨다.

- move\_points\_with\_center() 함수는 move\_points() 함수와 동일하게 모든 점들을 이동시키지만, 거기에 더해 무게중심에 해당하는 점도 같이 이동시키는 함수이다.

**(e) 이동 종료 시 색상 복귀**

- 구현 여부: 예

- 작동 확인 방법: 드래그 후 마우스에서 손을 떼면 선 색상이 원래의 파란색으로 돌아온다.

- 구현 방법:

- mousepress() 함수에 마우스 UP 이벤트 처리를 추가한다. 왼쪽 마우스 버튼 해제 시 st.leftbuttonpressed = 0, st.selected = 0으로 상태를 초기화한다.

- 이후 glutPostRedisplay() 함수를 호출하여 색상 복구를 반영한다.

**(f) 회전 모드에서도 이동 가능**

- 구현 여부: 예

- 작동 확인 방법: r키를 눌러 회전 모드 진입 후에도 중심점 드래그로 이동이 가능하다.

- 구현 방법:

- 기존 mousemove() 함수의 조건문에서 st.rotation\_mode 여부에 관계없이 polygon\_mode 상태일 경우에만 이동 가능하도록 조건을 조정한다. 이렇게 하면 회전모드에서도 무게중심 클릭 후 이동이 가능해진다.

- move\_points\_with\_center() 함수는 회전 중에도 호출 가능하도록 그대로 사용한다.

**(g) 마우스 휠로 확대/축소**

- 구현 여부: 예

- 작동 확인 방법: 다각형 완성 후, 마우스 휠을 위쪽 방향으로 스크롤 하면 다각형이 무게 중심을 중심으로 축소되고, 아래쪽 방향으로 스크롤 하면 무게 중심을 중심으로 확대된다.

- 구현 방법:

- 먼저, register\_callbacks() 함수 내부에 glutMouseWheelFunc(mousewheel); 코드를 추가한다. 해당 함수는 마우스 휠 입력을 감지해서 mousewheel() 콜백 함수로 이벤트를 전달한다. 휠을 위로 스크롤 하면 direction>0이 되고 아래로 스크롤 하면 direction<0이 된다.

- mousewheel() 함수에서는 매번 10%씩 확대/축소되도록 확대 계수를 로 설정한다. 축소 계수는로 설정한다. 이후 마우스 휠 스크롤 방향에 따라 zoom\_polygon() 함수에 해당하는 비율을 넘겨서 Scaling한다.

- zoom\_polygon() 함수에서는 각 점 (x, y)를 무게중심 (cx, cy) 기준으로 아래 방식으로 변환한다.

- 즉, 점을 중심으로 평행이동한 후 → 비율만큼 확대/축소 → 다시 되돌리는 구조이다. 이러한 구조는 사실상 **무게중심을 원점으로 이동 → Scaling → 원 위치 복귀**라는 ‘아핀 변환 구조’와 같다고 볼 수 있다.

**(h) 회전 모드에서도 확대/축소 가능**

- 구현 여부: 예

- 작동 확인 방법: r키로 회전 모드 진입 후에도 마우스 휠을 스크롤하면 다각형이 무게중심을 기준으로 확대 또는 축소된다.

- 구현 방법:

- mousewheel() 함수 내에는 st.rotation\_mode 또는 st.polygon\_mode와 관련된 별도의조건문이 없으며, 오직 다각형의 존재 여부에 기반해 확대/축소가 적용된다.

- 결과적으로, 회전 모드 (st.rotation\_mode == 1)가 활성화된 상태에서도 마우스 휠 이벤트는 정상적으로 처리되며, 기존 회전 애니메이션과 충돌 없이 병행 실행된다.