**컴퓨터 그래픽스 과제 1 Readme**

20171693 조병화

2022/03/30

**환경 명세**

OS – window10 64bit, Intel(R) Core(TM) i5-6200U CPU, intel HD Graphics 520 - win 64, release

**전체적인 프로그램의 구상**

처음 함수가 시행 되면, 입력으로 들어올 수 있는 사항들 (shift, 상하좌우 방향키, 왼쪽, 오른쪽 마우스, P키, R키, C키, F키) 들을 실습 파일의 예시를 따라 처리하였다.

shift 의 경우 GLUT\_ACTIVE\_SHIFT 를 이용하여 현재 눌려있는 상태인지 체크하여 shift를 누른상태에서만 일어나는 작업들을 구분하였다.

마우스 왼쪽 클릭, 마우스 오른쪽 클릭과 스크롤 하기의 경우 mousepress 에서 구분하여 주었다.

방향키 (상하좌우) 의 경우에는 다른 c,r,f 키와 구분하여 special 함수에서 처리해 주었다.

P,R,C,F 키의 경우 keyboard 함수에서 switch 문을 이용해 구분해 주었다.

각 입력들에 대해서 공통적으로 해당 입력이 있었다는 flag를 설정해 주고, glutPostRedisplay 를 통해 윈도우 창을 새로 그려주었다.

**display** 에선, 크게 다각형이 만들어 졌는지, 만들어 지지 않았는지로 나누어 만들어 지지 않았을 경우에는 GL\_POINTS로 점을 찍고, GL\_LINE\_STRIP 으로 그 점들을 선으로 이어서 출력해 주었다.

만약 다각형이 이미 완성된 상태에서 추가적인 입력이 들어왔을때에는 GL\_POINTS로 점을 찍고, GL\_LINE\_LOOP로 바로 다각형을 만들어 주었다. 이 때 점을 찍고, 다각형을 그리는 과정을 **draw\_func**라는 함수에서 해주고, display에선 해당 함수의 호출만을 담당하였다.

다음으론 각 입력의 세부 구현에 대한 설명이다.

**C키**가 입력으로 들어왔을 때에는 지금까지 입력된 점이 저장된 배열을 초기화하고, 모든 flag들 또한 초기화해주었다.

**R키**가 입력되었을 경우 기존의 draw\_func 함수 호출 인자를 달리하여 R키가 animation active 상태일 동안에는 무게중심 또한 함께 윈도우에 그려주었다.

**P키**가 입력되었을 경우 점 저장 배열에 몇 개의 점이 저장되었는지 판단 후 다각형 생성이 가능할 경우 생성해주고, 다각형 완성 flag를 세워주었다.

**F키**가 입력되었을 경우 프로그램을 종료해주었다.

**회전**의 경우 따로 만들어져있는 함수를 사용하지 않고 일정 radian 값을 설정하고 회전 변환을 저장된 모든 점들에 대해 적용시켜주었다. 이 작업은 calculate\_rotated() 함수를 통해 구현하였다.

우선 무게중심을 원점으로 이동시키며 그 평행이동 만큼 모든 점들도 함께 평행이동 시킨후, 회전 후 다시 원래의 위치로 점들을 다시 평행이동 시켜주었다.

**무게중심**을 출력해 주어야 했는데, 이 무게중심은 모든 점들의 x좌표의 합, y좌표의 합을 구하여 점의 개수로 나누어 구해주었다.

마우스 **오른쪽 scroll** 의 경우 현재 마우스의 위치를 이전의 위치와 비교하여 그 차이만큼 모든 점들을 평행이동 시켜주어 구현했다.

**각 요구사항 별 수행여부.**

모든 요구사항 구현을 성공하였다.

1. 초기 시행시 salmon 색의 초기 윈도우를 화면에 구현하는데 성공하였다.

ctrl – f5를 통해 실행시 바로 창이 뜬다.

1. shift 키를 누르고 있는 상황에서의 마우스 왼쪽 입력 시에만 점이 생성되도록 구현하는데 성공하였다.

shift를 누르고 왼쪽 마우스 클릭시 점이 찍힌다. shift 입력 판단 방식은 위에서 기술하였다.

1. 이 후 shift를 누른 상태에서 왼쪽 클릭으로 점을 생성할 때마다 이전 점과 이어주는 것을 구현하였다.

점들을 선들로 있는데 사용한 방법은 ‘구상’에서 기술하였다.

1. 3개 이상의 점과 그 점들이 연결 된 상태에서만 키보드 P 입력 시 다각형이 완성되는 것을 구현하였다.

다각형을 만드는데 사용한 방법은 ‘구상’에서 기술하였다.

1. C입력시 지금까지 윈도우에 그린 모든 점, 선분들을 지우는 기능을 구현하였다.

keyboard 함수에서 구현하였고, for문으로 모든 점들을 초기화하고 flag들도 초기화했다.

1. 점이 2개 이하일 때 p입력시 you need to choose more points to make a polygon 을 출력해주었다.

배열에 저장된 점 개수 확인 후 fprintf로 콘솔에 출력하였다.

1. 다각형 완성 시 왼쪽 클릭에 대한 입력을 무시하여 구현하였다.

flag를 설정하여 구현하였음.

1. 방향키 입력에 대한 이동을 좌표 평행이동을 이용해 구현하였고, 다각형 완성 flag 가 있을때에만 시행되게 하였다.

special 함수에서 이동 방향에 따라 모든 점들을 평행이동한 후 redisplay 해주었다.

1. 오른쪽 마우스 이동의 경우도 좌표의 평행이동을 통해 구현하였고, 마찬가지로 다각형 완성 flag 가 있을때에만 시행되게 하였다.
2. r키 입력시 회전변환을 이용해 회전하도록 구현하였고, 다각형 완성 flag 가 있을때에만 시행되게 하였다.

회전변환의 경우 animation mode 변수를 입력시마다 바꾸어 가며 gluttimerfunc 를 이용하여 구현하였다. 재귀 될 때마다 radian 값을 증가시켜 원래의 좌표에 대해 회전변환을 해준 후 그 좌표들을 기준으로 다시 redisplay 해주었다.

1. 윈도우에 다시 이미지를 찍는 함수에서 R키 입력 시, animated 일 때에만 활성화되도록 하여 무게중심을 출력해주었다.

위의 설명처럼 animation mode가 다시 해제되어 더 이상 rotate 하지 않도록 구현했다.

1. R을 회전 중 누를 시 animated 상태를 끄고 무게중심을 찍는 부분은 실행시키지 않았다.

draw\_func에 인자를 넣어 호출 시 animation mode 일때만 인자의 값을 달리해 무게중심을 출력해주었다.

1. r 입력시 설정되는 animation mode 에 따른 제한을 C(keyboard), 방향키(special), 마우스 스크롤(mousepress) 에 대해 모두 제한을 걸어주어 구현하였다.
2. 실습 예제 파일의 reshape 함수를 그대로 이용하였다.
3. f를 통해 프로그램 종료를 구현하였다.
4. 각자 다른 색깔을 적용하여 윈도우를 꾸미는데 성공하였다.