Mid Term 레포트

작성일	2022.04.27	전공	컴퓨터소프트웨어 공학
작성자	김주원	학번	20184009
강의명	컴퓨터그래픽스	강의자	홍 민 교수님

| 목차 |

1. Mid Term Project
2022년 중간 프로젝트

- Interactive하게 폴리곤을 그리는 프로그램 작성: 사용자가 매번 왼쪽 마우스 버튼을 누를 때마다 점들이 추가 되어야 하고, 오른쪽 마우스 버튼을 누르면 자동으로 마지막 라인을 추가하여 하나의 색으로 칠해진 폴리곤을 만든다. 이때 매번 마우스 클릭으로 점들이 추가될 때마다 점들이 보여야 하고, 점의 위에 해당 숫자가 나와야 한다. 처음 점이 0부터 시작해서 순서대로 증가한다. 항상 마지막 점으로부터 현재 점까지의 라인은 rubber band line으로 표현 되어야 한다. motion callback function, glutPostRedisplay function을 활용할 것
- 물체 전체를 드래그 하면 이후 마우스의 이동에 따라서 물체가 회전하면서 이동되어야 함
- 마우스의 Middle 버튼을 누르면 초기화되어야 함
- 4월 27일(수)자정 까지 작성해서 제출 하시오

1-1 문제분석

정점의 좌표를 기억할 자료구조를 만들어 마우스의 왼쪽클릭의 좌표를 저장하며 점을 출력한다. 정점의 위에는 정점의 번호가 출력되게 만들어준다. 이때 윈도우에 마우스가 올라갈 경우 마지막 정점이 마우스를 따라오도록 motion 콜백으로 만들고, 우클릭시에 해당 좌표의 정점이 마지막으로 삽입된 후에 저장된 좌표들을 불러와 폴리곤을 그린다.

마우스를 드래그하면 처음 클릭한 좌표와 드래그하며 이동한 마우스의 좌표를 motion 콜백을 이용하여 알아내 계속해서 드래그창을 그리도록 해준다. 이렇게 만들어진 드래그 창의 범위내에 폴리곤이 속하는지 확인하여 속하게 될 경우, idle 콜백함수로 마우스를 따라다니며 회전하게 만들어준다.

가운데 버튼을 누를 경우, 저장된 좌표를 전부 삭제하고 행렬을 초기화 시켜 다시 그림을 그릴 수 있도록 만들어주도록 해야 한다.

1-2 소스 코드 분석

```
/*
        프로그램 명 : 폴리곤 컨트롤 프로그램
        작성일 : 2020.04.18 ~ 04.27
        작성자 : 컴퓨터소프트웨어공학과 20184009 김주원
        프로그램 설명 : Interactive하게 폴리곤을 그리는 프로그램,
                        마우스의 좌클릭으로 정점을 찍고, 마지막 정점과
                        마우스 커서에 항상 rubber band line이 연결된다.
                        우클릭시에 생성한 정점들이 연결되어 폴리곤이 그려진다.
                        그려진 폴리곤을 드래그하면 폴리곤이 마우스를 따라다니며
                        회전하게 되고, 중앙 버튼을 누를 경우, 그려진 모든것이
                        초기화되며 다시 그릴수 있게 된다.
*/
#include <GL/glut.h>
#include <stdlib.h>
#include <stdio.h>
```

코드의 시작부분이다. 필요한 헤더파일과 주석으로 프로그램에 대한 설명을 작성하였다.

```
GLfloat spin = 0.0;
float width = 500;
float height = 500;     //행렬의 각도와 윈도우의 크기

GLint drag_minY, drag_maxY, drag_minX, drag_maxX;   //드래그 창의 범위
GLint SPx, SPy, EPx, EPy;   //드래그의 시작점과 마지막점
```

도형의 각도를 정하는 변수 spin 이다. 드래그를 하기 전에는 평범하게 그려야 하므로 0 으로 초기화 시켜주었다. width 와 height 는 윈도우의 크기를 설정하기위한 변수이다.

drag_min 변수는 드래그 창의 크기를 정하는 변수이다. SP 변수는 드래그시 마우스 포인터의 시작점이고 EP 는 마우스의 도착점이다.

```
float centerX = 0, centerY = 0;     //폴리곤의 중점
GLint polygon_minY, polygon_minX, polygon_maxY, polygon_maxX;   //폴리곤의 크기

int polygon = 0;            //폴리곤의 유무 판단 변수
int node_count = 0;         //정점의 개수 변수
int drag_activate = 0;      //드래그 활성 토글 변수
int selected_polygon = 0;   //폴리곤 드래그 판단 변수
```

마우스를 따라가기 위한 폴리곤의 중점 변수 center 와 폴리곤의 드래그 범위를 나타내는 변수 polygon 변수이다.

아래는 폴리곤의 활성화 확인 변수와 점의 개수, 드래그활성 여부 변수, 폴리곤이 드래그 범위에 속해져 드래그 됐는지 확인하는 변수들을 초기화 시켜주었다.

```
typedef struct  Node {      //정점의 구조체 정의
    struct Node *link;
    float vertex[2];
}node;

node *vertex_list = NULL;   //정점 리스트 변수 선언

node *addVertex(float x, float y);
void remove_list();                 //연결리스트 관련 함수들
```

폴리곤의 점의 좌표를 기억하기 위한 자료구조로 연결리스트를 선택하였다. 따라서 노드의 구조체를 정의하였고, 프로그램이 실행되는 동안 어느 함수에서나 접근이 쉽도록 전역변수로

선언해주었다. 아래는 연결리스트와 관련된 노드 추가함수와 연결리스트 공간 해제 함수들을 선언해준 것이다.

```
42  void init();                          //윈도우 관련 함수들
43  void reshape(int newWidth, int newHeight);
44  void display();
45
46  void mouseProcess(int button, int state, int x, int y);
47  void mouse_move(GLint x, GLint y);
48  void mouse_moveP(GLint x, GLint y); //마우스의 동작 관련 함수들
```

위의 세 함수들은 윈도우의 크기와 행렬의 초기화함수와 렌더 콜백함수를 선언해주었다. 아래 세 함수들은 마우스의 동작과 관련된 함수들이다. 마우스의 입력을 처리하는 콜백함수와 motionfunc 과 motionpassivefunc 의 콜백 함수들을 선언해준 것이다.

```
50  void MoveObject();
51  void spin_Display();                //도형 움직임 관련 함수들
52
53  void draw_points();
54  void draw_lines();
55  void draw_drag();
56  void draw_polygons();
57  void point_numbers();               //윈도우에 그릴 그림 관련 함수들
```

위의 두 함수들은 그림의 움직임과 관련된 함수이다. MoveObject 는 도형을 마우스를 따라가게 하며, spin_display 는 idlefunc 의 콜백함수로 그림의 각도를 계속해서 늘려 돌아가게끔 해주는 함수이다.

아래의 5 개 함수는 각각 저장된 좌표에 점 출력, 마지막 정점과 마우스를 잇는 선 출력, 드래그창 출력, 저장된 정점으로 이루어진 폴리곤, 정점의 번호를 출력해주는 함수들이다.

```
60  int main(int argc, char ** argv) {
61      glutInit(&argc, argv);      //GLUT초기화
62      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);    //단일 버퍼 창과 GLUT의 기본 색상 모드
63      glutInitWindowSize(width, height);      //윈도우 사이즈를 설정
64      glutInitWindowPosition(150, 150);       //윈도우 생성 위치 설정
65      glutCreateWindow("Interactive Drawing Polygon");        //윈도우를 생성한다
66      init();                     //윈도우창을 초기화한다
```

메인 함수이다. 맨 처음 glutInit 로 GLUT 라이브러리를 초기화하고, 운영체제와 연결시켰다. 그리고 애니메이션이 단순하므로 싱글 버퍼를 사용하고, 미리 초기화 시켜 놓은 변수로 윈도우의 크기를 설정하였다. 윈도우의 생성위치도 설정해주었다. 그리고 정의된 함수로 윈도우의 행렬등을 초기화 시킨다.

```
68      glutDisplayFunc(display);       //display함수를 디스플레이 이벤트 콜백 함수로 등록
69      glutReshapeFunc(reshape);       //변경된 렌더링 크기로 윈도우 크기 재설정
70      glutMouseFunc(mouseProcess);    //mouseProcess함수를 마우스 이벤트 콜백 함수로 등록
71      glutPassiveMotionFunc(mouse_moveP);     //윈도우 위에서 움직이는 마우스의 움직임 처리 함수
72      glutMotionFunc(mouse_move);     //마우스의 클릭상태의 움직임 처리 함수
73      glutMainLoop();     //이벤트 루프로 진입
74      remove_list();
75      free(vertex_list);
76      return 0;
77  }
```

윈도우에 무언가를 그리기 위한 display 함수로 콜백 함수를 설정해주었고, 윈도우의 크기가 변함에 따라 다시 윈도우를 설정해주는 함수로 reshape 함수를 콜백 함수로 사용하였다.

마우스의 입력을 받아 특정 동작을 하는 콜백함수로 mouseProcess 함수를 설정하였다. 마우스가 윈도우 위에 있을 때의 동작 함수를 mouse_moveP 로 설정하였다. 또 마우스가

윈도우 위에서 클릭상태와 동시에 움직일 때의 동작 함수를 mouse_move 로 설정하였다. 출력이 끝나고 연결리스트가 삭제되도록 연결리스트를 해제하도록 했는데, 실제로는 출력창을 닫게 되면 프로그램이 그대로 종료되기 때문에 실행되지 않는 부분이다. 쓸데없이 추가된 부분이라는 걸 알게 되었다.

```
79  void init() {
80      glClearColor(0.0f, 0.0f, 0.0f, 0.0f);   //배경 색상 설정
81      glMatrixMode(GL_PROJECTION);        //투영 행렬 모드로 변환
82      glLoadIdentity();       //행렬을 단위 행렬로 만듦
83      gluOrtho2D(0.0, width, height, 0.0);    //행렬을 볼 영역을 선택
84  }
```

윈도우를 초기화하는 함수이다. 배경의 색상을 특정 색상으로 설정해주었고, 현재 윈도우를 투영행렬모드로 설정하고 단위행렬로 만든 후에 오른쪽과 위의 좌표계로 그림을 그리기 위해 윈도우의 크기로 right, top 의 인자로 값을 넘겨 매트릭스를 설정하였다.

```
86  void reshape(int newWidth, int newHeight) { //윈도우 크기 만큼 렌더링 화면 비율을 바꾸기 위한 함수
87      glViewport(0, 0, newWidth, newHeight);      //랜더링 영역 설정
88      glMatrixMode(GL_PROJECTION);    //모델뷰 행렬을 가져온다
89      glLoadIdentity();   //단위행렬로 초기화
90      gluOrtho2D(0, newWidth, newHeight, 0);  //변경된 윈도우의 크기로 볼 행렬 설정
91      glutPostRedisplay();    //바뀐 윈도우 재생(새로고침)
92  }
```

도형의 크기를 재설정하는 reshape 콜백 함수의 함수이다. 변한 윈도우의 크기로 뷰포트를 재설정하고, 다시 투영행렬모드(GL_PROJECTION)로 설정하고 단위행렬로 바꾼 후에 새로운 윈도우의 크기의 좌표계를 보게끔 설정하였다. 마지막으로 바뀐 화면을 출력한다.

```
94  void display() {    //Display콜백 함수
95      glClear(GL_COLOR_BUFFER_BIT);       //저장된 색으로 화면을 초기화
96      glMatrixMode(GL_MODELVIEW);         //모델뷰 행렬을 가져옴
97      glLoadIdentity();                   //바뀐 윈도우를 재생시킴(새로고침)
98
99      if (selected_polygon == 1) {    //폴리곤이 드래그로 선택된 경우,
100         glutIdleFunc(spin_Display); //각도를 계속 변경
101         MoveObject();       //변경된 각도 상태로 특정 위치로 이동
102     }
```

미리 지정해둔 색상으로 화면을 출력 후 모델 좌표계와 시점 좌표계의 공간을 계산하기 위해 GL_MODELVIEW 로 설정하고 단위행렬로 바꾸어 주었다.

이제 그림을 그리기 시작한다. 만약 폴리곤이 드래그 됐다면 도형을 돌리기 위해 idle 콜백으로 각도를 계속 바꿔주고, 바뀐 각도로 마우스의 좌표에 그림을 그리도록 한다.

```
104     glColor3f(0, 0, 1);
105     //폴리곤 출력, 폴리곤 미생성시 출력x
106     if (polygon == 1)
107         draw_polygons();
108
109     glColor3f(1, 1, 1);
110     draw_points();      //저장된 정점의 위치에 점 출력
111
112     //선 출력, 정점 미생성시, 폴리곤 생성시 출력x
113     if (vertex_list != NULL && polygon == 0)
114         draw_lines();
```

폴리곤의 색상을 설정하고 폴리곤 생성 상태라면 폴리곤을 출력한다. 다음으로 색을 바꾸고 저장된 좌표에 점을 출력한다. 또 정점이 저장 돼있고, 폴리곤 미생성 상태라면 마지막 정점과 마우스를 연결하는 선을 출력하도록 한다.

```
116         //드래그 창 출력, 폴리곤 미생성시, 우클릭시, 폴리곤 선택시 출력x
117         if (polygon == 1 && drag_activate == 1 && selected_polygon == 0)
118             draw_drag();
119
120         point_numbers();    //점 위에 숫자 출력
121
122         glFlush();      //그려진 것들 출력
123     }
```

폴리곤이 생성됐고, 드래그 활성화 변수가 1 이며, 선택된 폴리곤이 없다면 드래그를 그릴 수 있도록 한다. 그리고 정점의 위치에 정점의 번호를 출력하며 마지막으로 모아둔 그림들을 전부 내보내 그림이 출력된다.

```
125 void mouseProcess(int button, int state, int x, int y) {    //마우스 동작 처리 함수
126     switch (button) {
127     case GLUT_LEFT_BUTTON:
128         if (state == GLUT_DOWN) {   //왼쪽 버튼 클릭시
129             if(polygon == 1)        //폴리곤 생성시,
130             drag_activate = 1;      //드래그 활성화
131             SPx = x, SPy = y;       //드래그의 시작점
132             if (polygon == 0) {     //폴리곤 미생성시,
133                 vertex_list = addVertex(x, y);  //새로운 정점 리스트에 삽입
134             }
135             glutPostRedisplay();    //바뀐 윈도우 재생(새로고침)
136             break;
137         }
```

마우스의 콜백을 처리하는 함수이다. 좌클릭이 됐을 경우, 폴리곤이 그려져 있다면, 드래그가 활성화된다. 현재 클릭한 좌표를 드래그의 시작점으로 정하고, 폴리곤이 없다면 리스트에 좌표를 저장한다. 마지막으로 바뀐 윈도우를 재생한다.

```
138         else if (state == GLUT_UP) {    //클릭후 올라왔을 때
139             if (drag_minX <= polygon_minX && drag_maxX >= polygon_maxX &&
140                 drag_minY <= polygon_minY && drag_maxY >= polygon_maxY)
141                 selected_polygon = 1;   //드래그 범위내 폴리곤 속할시, 선택 상태로 변환
142             drag_activate = 0;      //드래그 비활성화
143             break;
144         }
```

좌클릭 후에 마우스가 올라오면 드래그가 끝난 상태이다. 왼쪽 마우스 버튼이 올라왔을 때 드래그 영역에 폴리곤이 속한다면 선택된 상태로 변수를 설정한다. 그리고 드래그 창이 다시 안 그려지게 하기 위해 드래그 활성화 변수를 0 으로 설정한다.

```
145     case GLUT_RIGHT_BUTTON:
146         if (state == GLUT_DOWN) {   //오른쪽 버튼 클릭시
147             if (polygon == 0) {     //폴리곤 미생성시
148                 polygon = 1;        //폴리곤 생성 표시
149                 vertex_list = addVertex(x, y);  //마지막 정점 좌표 리스트에 삽입
150                 centerX /= node_count, centerY /= node_count;   //정점들의 중점 계산
151             }
152             glutPostRedisplay();    //바뀐 윈도우 재생(새로고침)
153             break;
154         }
```

우클릭이 될 경우, 폴리곤이 그려진다. 이때 폴리곤이 그려져 있다면 다시 그릴 필요가 없으므로 폴리곤이 안 그려진 상태를 나타내는 변수 polygon 이 0 일때의 조건을 넣었다. 따라서 안 그려졌을 때 polygon 변수를 1 로 바꾸고 우클릭시의 좌표도 점으로 추가해야 하므로 리스트에 좌표를 삽입하였고 현재까지 추가된 정점의 개수로 중점을 계산하였다. 마지막으로 바뀐 윈도우를 재생한다.

```
155     case GLUT_MIDDLE_BUTTON:
156         if (state == GLUT_DOWN) {       //가운데 버튼 클릭시, 초기화
157             polygon = 0;                //폴리곤이 미생성된 상태
158             node_count = 0;             //정점 개수 0
159             selected_polygon = 0;       //선택된 폴리곤이 없는 상태
160             drag_activate = 0;          //드래그 비활성화
161             centerX = 0, centerY = 0;   //중점 0
```

가운데 버튼이 눌린다면 초기화를 시켜준다. 폴리곤을 비활성화하고, 노드의 개수를 0 으로, 폴리곤의 선택 여부를 0 으로, 드래그를 비활성화하고, 계산한 중점을 0 으로 초기화한다.

```
162             //남아있을 드래그 범위 초기화
163             drag_minY = 0, drag_maxY = 0, drag_minX = 0, drag_maxX = 0;
164             remove_list();              //정점 리스트 삭제
165
166             spin = 0.0;                 //각도 초기화
167             glutIdleFunc(NULL);         //계속 동작하는 idle콜백 함수 정지
168
169             init();                     //윈도우 초기화
170             glutPostRedisplay();        //바뀐 윈도우 재생(새로고침)
171             break;
172         }
173     }
174 }
```

처음 드래그한 범위의 값을 0 으로 초기화하고 리스트에 저장된 좌표를 전부 삭제한다. 바뀐 각도를 0 으로 초기화하고, 작동되던 idle 콜백 함수를 없애준다.

```
176 void mouse_move(GLint x, GLint y) {     //마우스 클릭후 이동함수
177     EPx = x; EPy = y;
178     glutPostRedisplay();    //바뀐 윈도우 재생(새로고침)
179 }
180
181 void mouse_moveP(GLint x, GLint y) {    //마우스 미클릭 이동함수
182     EPx = x; EPy = y;
183     glutPostRedisplay();    //바뀐 윈도우 재생(새로고침)
184 }
```

위의 함수는 motionfunc 의 콜백 함수이다. 클릭하며 이동할 때 바뀌는 좌표를 계속해서 EP 변수에 저장해주며 바뀌는 화면을 재생시킨다.

아래는 Passivemotionfunc 의 콜백 함수이다. 윈도우 위에서 움직이는 마우스의 좌표를 계속해서 EP 변수에 저장해주며 바뀌는 화면을 재생시킨다.

```
186 void MoveObject() {      //폴리곤의 중점을 마우스로 이동, 회전 함수
187     glTranslatef(EPx, EPy, 0.0);     //3.회전된 오브젝트 마우스 위치로 이동
188     glRotatef(spin, 0.0, 0.0, 1.0); //2.좌표계의 원점에서 spin각도만큼 회전
189     glTranslatef(-centerX, -centerY, 0.0);  //1.폴리곤 중점 좌표계 원점으로 이동
190 }
```

그림을 이동시키며 회전시키는 함수이다. 그림이 제자리에서 회전하기 위해 좌표계의 원점으로 이동시킨 후에 계속해서 증가하는 spin 변수의 각도로 그림을 돌려줘 회전하는 것처럼 만들어준다. 그리고 회전한 그림을 마우스의 좌표로 다시 이동시켜 마우스를 따라가게 만들어준다.

```
192 void spin_Display() {        //각도 변환 함수
193     spin = spin + 0.05;      //0.05씩 각도 변환
194     if (spin > 360)          //한바퀴를 돌았다면
195         spin = spin - 360;   //다시 0도부터 회전
196     glutPostRedisplay();     //바뀐 윈도우 재생(새로고침)
197 }
```

그림의 각도 값인 spin 을 계속해서 증가시켜 돌아가게 만드는 함수이다. 0.05 씩 회전하고 360 도를 넘을 경우에 다시 0 도부터 시작한다. 바뀐 윈도우를 재생시킨다.

```
199 void draw_points() {    //정점 출력 함수
200     node *vertex_head = NULL;
201     glPointSize(4);
202     glBegin(GL_POINTS);
203     for (vertex_head = vertex_list; vertex_head != NULL; vertex_head = vertex_head->link){
204         glVertex2fv(vertex_head->vertex);   //리스트에 저장된 좌표에 점 출력
205     }
206     glEnd();
207 }
```

리스트에 저장된 좌표에 점을 출력하는 함수이다. 연결리스트의 노드를 순회하여 저장된 좌표에 점을 전부 출력시킨다.

```
209 void draw_lines() {     //마지막 정점과 마우스 커서 러버밴드 연결
210     glBegin(GL_LINES);
211     glVertex2fv(vertex_list->vertex);
212     glVertex2f(EPx, EPy);       //가장 마지막 정점과 마우스 좌표 연결선 출력
213     glEnd();
214 }
```

motionfunc 으로 계속해서 바뀌는 마우스의 좌표와 리스트의 맨 처음 좌표(마지막에 추가된 정점)를 잇는 선을 출력하는 함수이다.

```
216 void draw_polygons() {  //저장된 정점으로 폴리곤 출력
217     node *vertex_head = NULL;
218     glBegin(GL_POLYGON);
219     for (vertex_head = vertex_list; vertex_head != NULL; vertex_head = vertex_head->link)
220         glVertex2fv(vertex_head->vertex);
221     glEnd();
222 }
```

리스트에 저장된 정점으로 이루어진 폴리곤을 그리는 함수이다. 리스트의 노드를 순회하여 저장된 좌표로 이루어진 폴리곤을 그린다.

```
224 void point_numbers() {
225     node *vertex_head = NULL;
226     int i = node_count, n = 0;
227     char msg1[3];   //세자리 수까지 저장 가능
228     glColor3f(1, 1, 1);//두자리 수까지
229     if (vertex_list != NULL) {
230         for (vertex_head = vertex_list; vertex_head != NULL; vertex_head = vertex_head->link) {
231             i--;    //노드가 5개일 경우 5, 4, 3, 2, 1, 0의 순서로 출력한다.
232             glRasterPos2fv(vertex_head->vertex);    //출력할 문자의 위치를 정함
233             sprintf_s(msg1, "%d", i);   //정수를 문자로 변환
234             for (n = 0; n <= i / 10; n++)   //자릿수를 계산하여 자릿수만큼 출력
235                 glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, msg1[n]);
236         }
237     }
238 }
```

점의 좌표에 점의 순서를 출력하는 함수이다. 리스트에 좌표의 순서가 거꾸로 되어있기 때문에 노드의 개수를 가져와 하나씩 감소하여(점이 5 개일 경우 5, 4, 3, 2, 1) 출력된다. 먼저 번호를 매길 정점의 좌표를 가져와 숫자가 출력될 위치를 설정하고 숫자가 문자형으로

저장(2 자리 수까지 저장 가능)돼 있기 때문에 해당 위치의 번호의 자릿수를 계산하여 반복하여 자릿수만큼 문자를 출력한다.

```
240  void draw_drag() {
241      //마우스의 시작점과 끝점의 좌표로 드래그창 생성
242      glBegin(GL_LINES);
243      glVertex2f(SPx, SPy);
244      glVertex2f(SPx, EPy);
245
246      glVertex2f(SPx, EPy);
247      glVertex2f(EPx, EPy);
248
249      glVertex2f(EPx, EPy);
250      glVertex2f(EPx, SPy);
251
252      glVertex2f(EPx, SPy);
253      glVertex2f(SPx, SPy);
254      glEnd();
```

드래그 창을 그리는 함수이다. 마우스의 시작 좌표와 현재 좌표의 x 축, y 축을 이용하여 선을 4 개를 그려 드래그창의 형태를 그린다.

```
255      if (SPx > EPx) {
256          drag_maxX = SPx;
257          drag_minX = EPx;
258      }
259      else {
260          drag_maxX = EPx;
261          drag_minX = SPx;
262      }
263
264      if (SPy > EPy) {
265          drag_maxY = SPy;
266          drag_minY = EPy;
267      }
268      else {
269          drag_maxY = EPy;
270          drag_minY = SPy;
271      }//시작점과 끝점의 좌표로 드래그 범위 도출
272  }
```

드래그의 선택 영역 값을 저장하는 부분이다. 시작점과 끝점의 x, y 축값을 계산하여 더 큰 값을 drag_max 에 더 작은 값은 drag_min 에 저장한다. 이렇게 저장된 값을 이용하여 폴리곤이 드래그 영역에 포함되는지 계산한다.

```
274  node *addVertex(float x, float y) { //리스트의 맨 처음에 노드를 추가하는 함수
275      node *newNode = (node*)malloc(sizeof(node));
276      newNode->vertex[0] = x;      //x축 값
277      newNode->vertex[1] = y;      //y축 값
278      newNode->link = vertex_list;
279      //추가되는 좌표값을 폴리곤 중점 변수에 합함
280      centerX += x; centerY += y;
281      if (vertex_list == NULL) {  //첫 정점으로 폴리곤의 범위 초기화
282          polygon_minY = y, polygon_minX = x, polygon_maxY = y, polygon_maxX = x;
283      }
```

리스트에 좌표를 삽입하는 함수이다. x, y 좌표값을 매개변수로 가져와 노드의 포인터 형으로 동적할당을 한 후에 좌표값을 저장하고 새로 생성된 노드의 링크로 기존의 연결리스트의 시작점(헤더)를 연결한다. 이때 좌표가 추가될 때마다 좌표값을 center 값에 더해주어 폴리곤을 그릴 때 폴리곤의 중점을 계산할 수 있도록 한다.

여기서 폴리곤의 드래그 활성 영역을 결정하는 동작이 일어난다. 드래그창은 사각형이기 때문에 폴리곤의 가장 바깥 점들의 값들로 선택될 영역을 정한다. 첫 정점이 들어갈 때는 맨 처음 정점의 좌표값으로 초기화 한다.

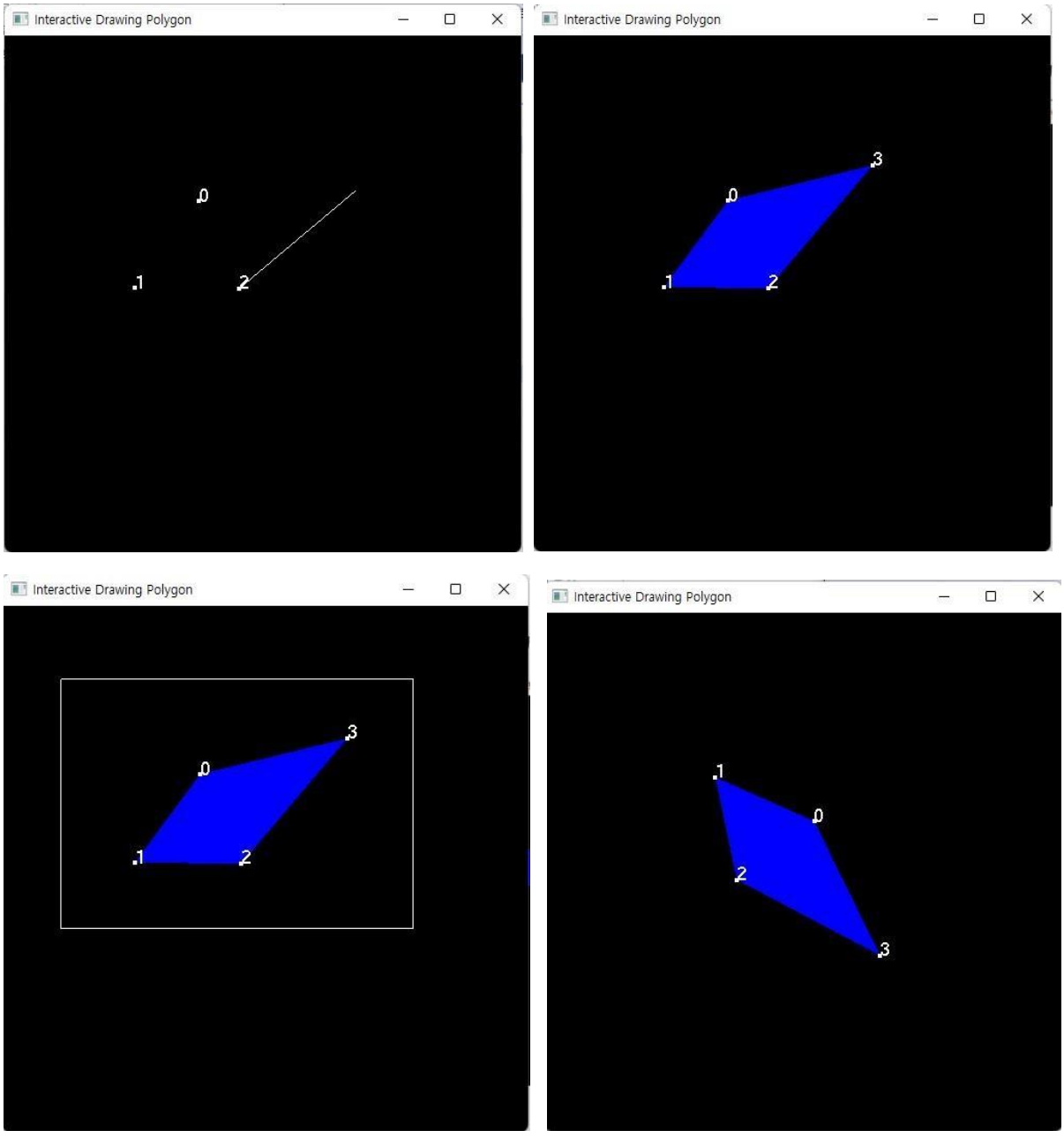
```
284     else {  //추가되는 정점이 기존의 범위를 넘어가면 다시 초기화
285         if (x > polygon_maxX)
286             polygon_maxX = x;
287         else if (x < polygon_minX)
288             polygon_minX = x;
289 
290         if (y > polygon_maxY)
291             polygon_maxY = y;
292         else if (y < polygon_minY)
293             polygon_minY = y;
294     }
295     vertex_list = newNode;
296     node_count++;       //노드 개수 +1
297     return vertex_list; //정점이 추가된 리스트 반환
298 }
```

추가되는 좌표가 값이 기존의 영역을 벗어나게 되면, 작고 큼에 따라 각각 polygon_min(가장 작은 좌표값) 또는 polygon_max(가장 큰 좌표값)에 저장되게 된다. 마지막으로 추가된 좌표의 노드를 헤더로 만들고 반환해준다. 결론적으로 연결리스트의 헤더에 노드를 삽입하는 동작의 함수이다. 좌표가 추가될 때마다 정점의 개수를 +1 해준다.

```
300 void remove_list() {    //연결리스트 삭제 함수
301     node *temp = NULL;
302     while (vertex_list != NULL) {
303         temp = vertex_list->link;
304         free(vertex_list);
305         vertex_list = temp;
306     }
307 }
```

마우스 가운데 버튼을 누르면 초기화를 시키는 동작이 일어난다. 따라서 리스트를 삭제하는 함수이다. 현재 리스트의 다음 노드를 가져와 임시로 저장하고 현재의 노드의 메모리를 해제하고 임시로 저장한 다음 노드를 다시 가져와 메모리 해제를 이어나간다.

1-3 실행창


Interactive Drawing Polygon
0
1
2
Interactive Drawing Polygon
3
0
1
2
Interactive Drawing Polygon
3
0
1
2
Interactive Drawing Polygon
1
0
2
3

1-4 느낀 점

자료구조를 연결리스트로 선택하였는데 연결리스트의 단점이 체감이 되는 과제였다. 매 동작시에 연결리스트를 접근하여 순회해야 하는데, 접근할 때 코드가 꽤 복잡하여 혼란스러웠다.

reshape 함수를 구현하여 윈도우의 크기가 변할 때를 대비하였는데, 윈도우의 크기를 바꾸고 난 후에 마우스 가운데 버튼을 클릭하여 초기화를 할 경우에 reshape 함수가 동작하지 않아 출력이 비정상적으로 일어나게 돼 아쉬웠다. OPENGL 코드는 연속적으로 일어나는 그림을 출력하는 프로그램이기 때문에 어떤 변화가 일어날 때 변화에 대응할 수 있도록 코드를 작성하는 것이 매우 중요한 것 같다.