컴퓨터 그래픽스

과제 3

학과 : AI 컴퓨터공학부

학번 : 201811302

이름 : 손현태

제출 날짜 : 2022-11-16

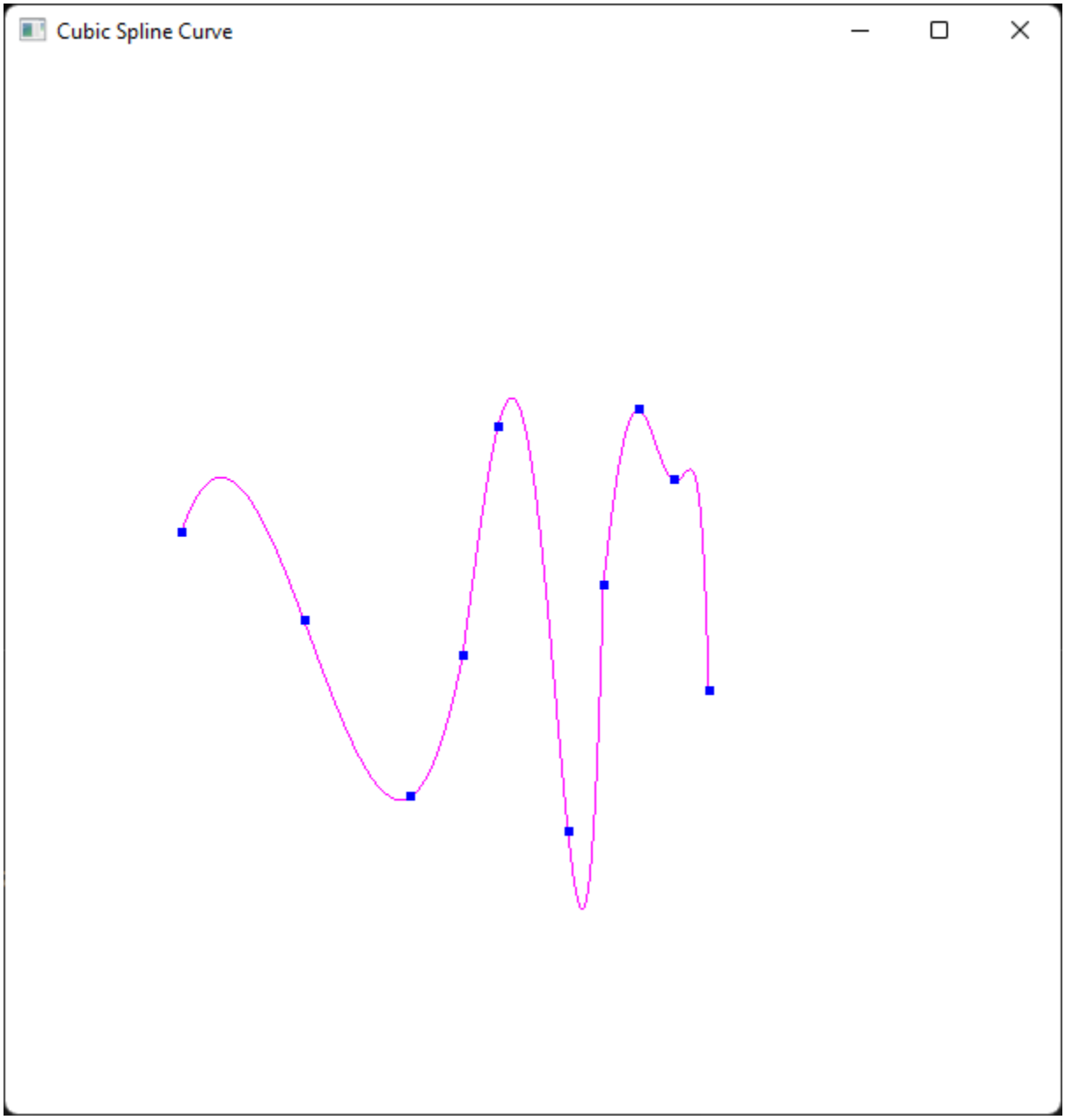
1. 임의의 Cotnrol Point를 통과하는 Cubic Spline Curve(3차 스플라인 곡선)를 그려보시오. 단 최소한 3개 이상의 구간별 Cubic spline Curve가 연결되는 커브를 설계하고 그려보시오. (한 개의 구간별 커브를 그리려면 각각 4개의 제어점이 필요하다. 따라서 2개의 구간별 커브는 공통점 1개 등 최소 7개의 제어점이 필요하며, 3개의 구간별 커브를 연결하기 위해 최소 10개의 제어점이 필요함. 제어점의 위치는 각자 제시하시오.)

P01.cpp

P01 (Global Scope)

```cpp
#include <iostream>
#include <GL/glut.h>

GLfloat ctrlPoints[10][3] = {
    {-20, 3, 0}, {-13, -2, 0}, {-7, -12, 0},
    {-4.0, -4.0, 0.0}, {-2.0, 9.0, 0.0},
    {2.0, -14.0, 0.0}, {4.0, 0.0, 0.0},
    {6, 10, 0}, {8, 6, 0}, {10, -6, 0}
};

void reshape(int w, int h) {
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-30.0, 30.0, -30.0, 30.0, -30, 30);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

float poly(float points[][3], float x, int n) {
    float y;
    float num = 1.0, d = 1.0;
    float sum = 0.0;

    for (int i = 0; i < n; ++i) {
        num = d = 1.0;

        for (int j = 0; j < n; ++j) {
            if (j == i) {
                continue;
            }

            num = num * (x - points[j][0]);
        }

        for (int j = 0; j < n; ++j) {
            if (j == i) {
                continue;
            }

            d = d * (points[i][0] - points[j][0]);
        }
        sum += num / d * points[i][1];
    }

    y = sum;

    return y;
}
```

```
50
51  void display(void) {
52      int i;
53      float x, y;
54
55      glClear(GL_COLOR_BUFFER_BIT);
56      glColor3f(1.0, 0.0, 1.0);
57      glBegin(GL_LINE_STRIP);
58
59      for (i = -200; i <= -40; i++) {
60          x = (float)i / 10.0;
61          y = poly(ctrlPoints, x, 4);
62          glVertex2f(x, y);
63      }
64
65      for (i = -40; i <= 40; i++) {
66          x = (float)i / 10.0;
67          y = poly(ctrlPoints, x, 7);
68          glVertex2f(x, y);
69      }
70
71      for (i = 40; i <= 100; i++) {
72          x = (float)i / 10.0;
73          y = poly(ctrlPoints, x, 10);
74          glVertex2f(x, y);
75      }
76
77      glEnd();
78      glBegin(GL_LINE_STRIP);
79
80      glEnd();
81      glPointSize(5.0);
82      glColor3f(0.0, 0.0, 1.0);
83      glBegin(GL_POINTS);
84
85      for (i = 0; i < 10; i++) {
86          glVertex3fv(&ctrlPoints[i][0]);
87      }
88
89      glEnd();
90      glFlush();
91  }
92
93  int main(int argc, char** argv) {
94      glutInit(&argc, argv);
95      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
96      glutInitWindowSize(600, 600);
97      glutInitWindowPosition(500, 300);
98      glutCreateWindow("Cubic Spline Curve");
99      glClearColor(1.0, 1.0, 1.0, 1.0);
100     glutDisplayFunc(display);
101     glutReshapeFunc(reshape);
102     glutMainLoop();
103
104     return 0;
105 }
```

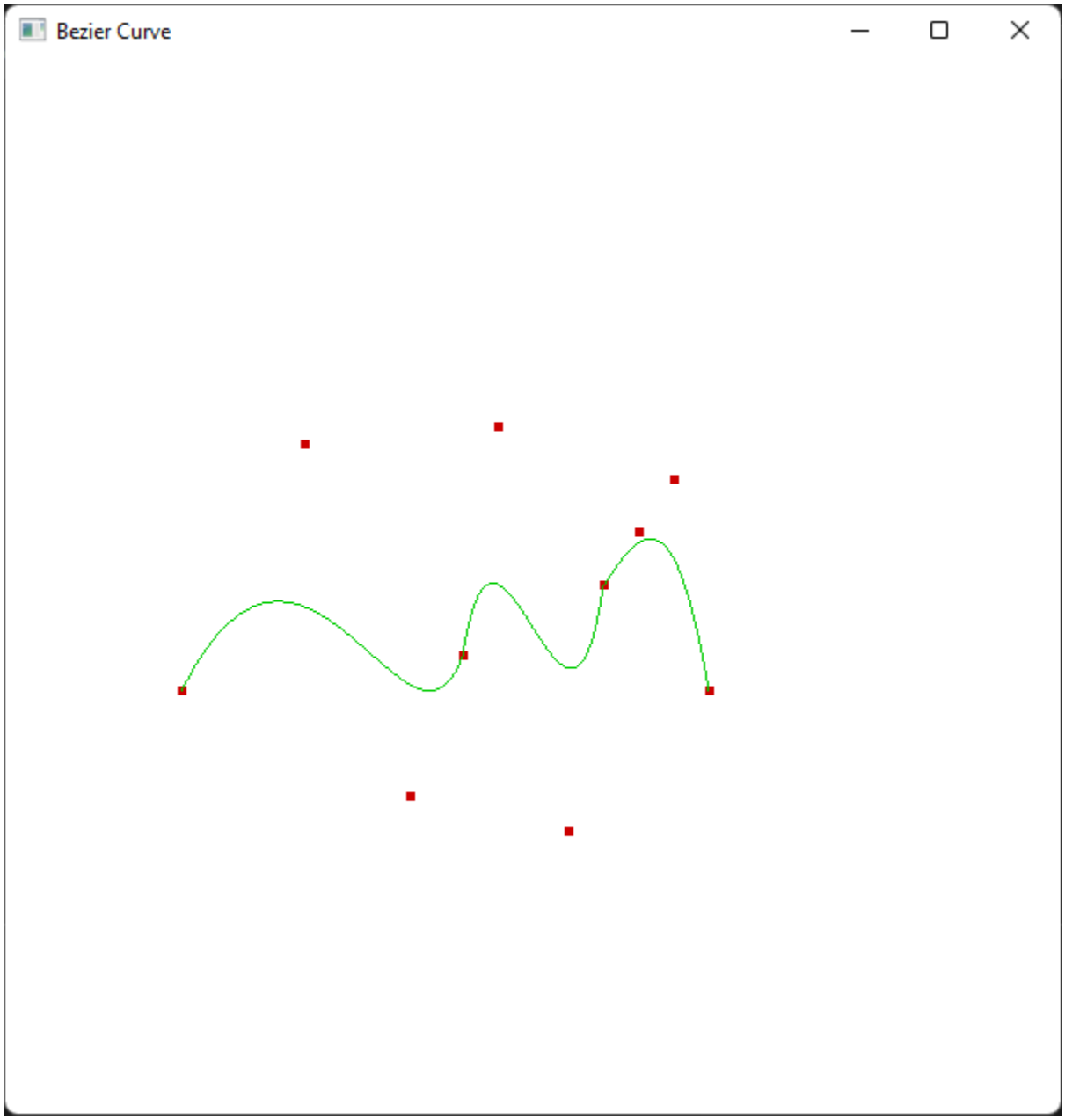

Cubic Spline Curve

2. 임의의 Control Point가 주어졌을 때 이 점들에 근사하는 Bezier Curve를 그려보시오. 반드시 최소한 두 개 이상의 구간별 Bezier Curve가 연결되는 커브를 설계하고 그려보시오. (한 개의 구간별 커브를 그리려면 4개의 제어점이 필요, 따라서 2개의 구간별 커브는 공통점 1개 등 최소 7개의 제어점이 필요하며 ,3개 구간별 커브를 연결하기 위해 최소 10개의 제어점이 필요함. 제어점의 위치는 각자 제시하시오.)

P02.cpp

P02 (Global Scope)

```cpp
#include <iostream>
#include <GL/glut.h>

GLfloat ctrlPoints[10][3] = {
    {-20, -6, 0}, {-13, 8, 0}, {-7, -12, 0},
    {-4.0, -4.0, 0.0}, {-2.0, 9.0, 0.0},
    {2.0, -14.0, 0.0}, {4.0, 0.0, 0.0},
    {6, 3, 0}, {8, 6, 0}, {10, -6, 0}
};

void reshape(int w, int h) {
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-30.0, 30.0, -30.0, 30.0, -30, 30);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glPointSize(5.0);
    glColor3f(0.8, 0.0, 0.0);
    glBegin(GL_POINTS);
    for (int i = 0; i < 10; i++) {
        glVertex3fv(&ctrlPoints[i][0]);
    }
    glEnd();

    for (int i = 0; i <= 6; i = i + 3) {
        glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4, &ctrlPoints[i][0]);
        glEnable(GL_MAP1_VERTEX_3);

        glColor3f(0.0, 0.8, 0.0);
        glBegin(GL_LINE_STRIP);

        for (int j = 0; j <= 30; j++) {
            glEvalCoord1f((GLfloat)j / 30.0);
        }
        glEnd();
    }

    glFlush();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(600, 600);
    glutInitWindowPosition(500, 300);
    glutCreateWindow("Bezier Curve");
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();

    return 0;
}
```

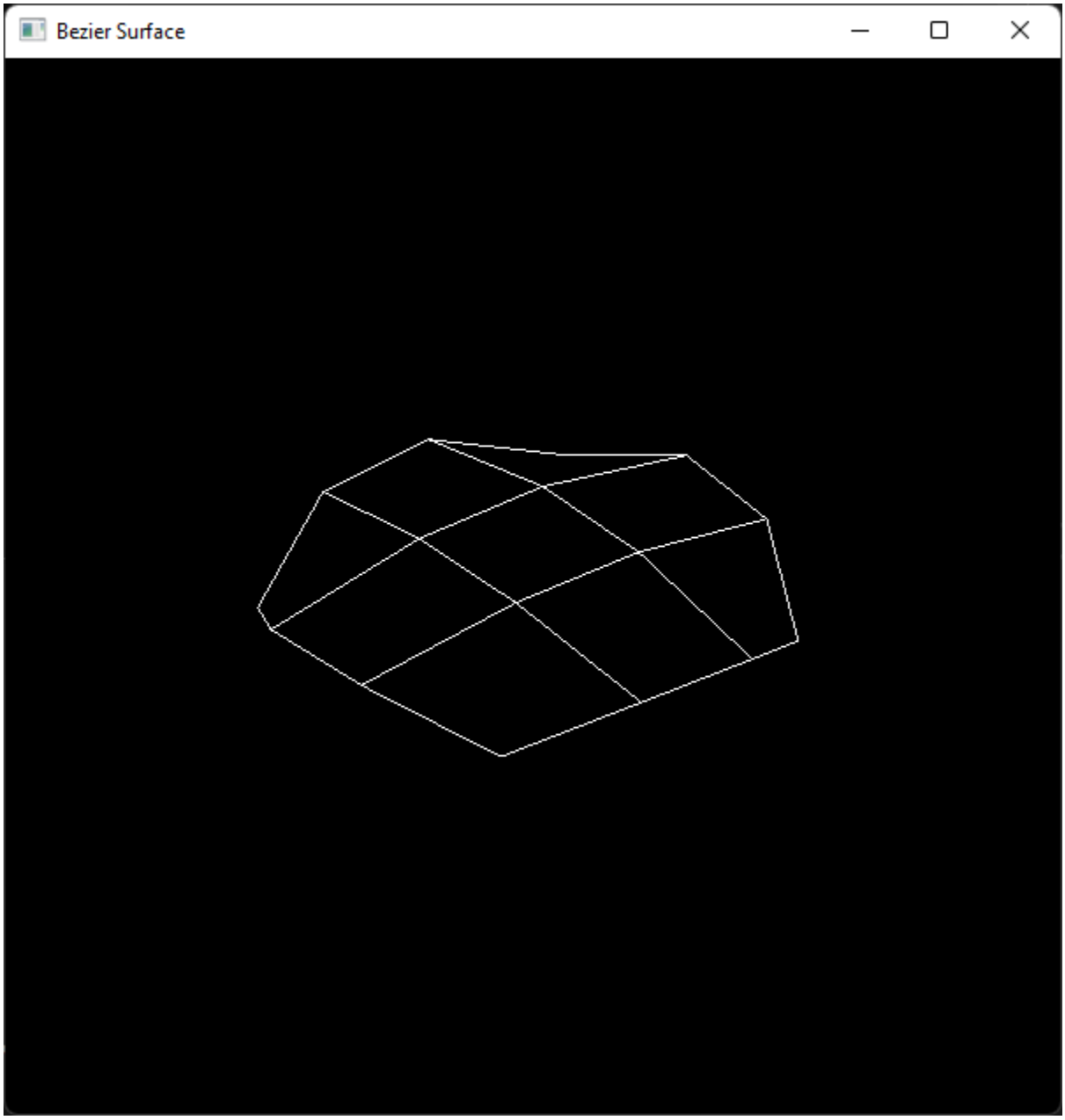

Bezier Curve

3. 아래 그림과 같이 Bezier Surface를 생성하기 위한 임의의 제어점을 제시하고 설계하고 설계된 제어점에 따른 Bezier Surface를 그리시오.

P03.cpp

P03 (Global Scope)

```
#include <GL/glut.h>
#include <iostream>
#include <windows.h>
#include <math.h>

GLfloat ctrlpoints[4][4][3] = {
    { {0,0 ,-1 }, {-0.5,0.24 ,-0.87 }, {-0.87,0.24 , -0.5}, {-1, 0, 0}},
    { {0.5,0.24 ,-0.87 },{0,0.33 ,-0.19 },{-0.19,0.33 ,0 }, {-0.87, 0.24, 0.5}},
    { {0.87,0.24 ,-0.5 }, {0.19, 0.33, 0},{0, 0.33,0.19 }, {-0.5,0.24 ,0.87 }},
    { {1,0 ,0 }, {0.87, 0.24,0.5 },{0.5, 0.24,0.87}, {0,0 ,1 }}
};

void reshape(int w, int h) {
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-2.0, 2.0, -2.0, 2.0, -2.0, 2.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void display(void) {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glRotatef(30, 1.0, 0.0, 0.0);
    glRotatef(-7, 0.0, 1.0, 0.0);
    glEvalMesh2(GL_LINE, 0, 3, 0, 3);
    glPopMatrix();
    glFlush();
}

void main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(600, 600);
    glutInitWindowPosition(500, 300);
    glutCreateWindow("Bezier Surface");
    glMap2f(GL_MAP2_VERTEX_3, 0, 1, 3, 4, 0, 1, 12, 4, &ctrlpoints[0][0][0]);
    glEnable(GL_MAP2_VERTEX_3);
    glMapGrid2f(3.0, 0.0, 1.1, 3.0, 0.0, 1.1);
    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutMainLoop();
}
```


Bezier Surface