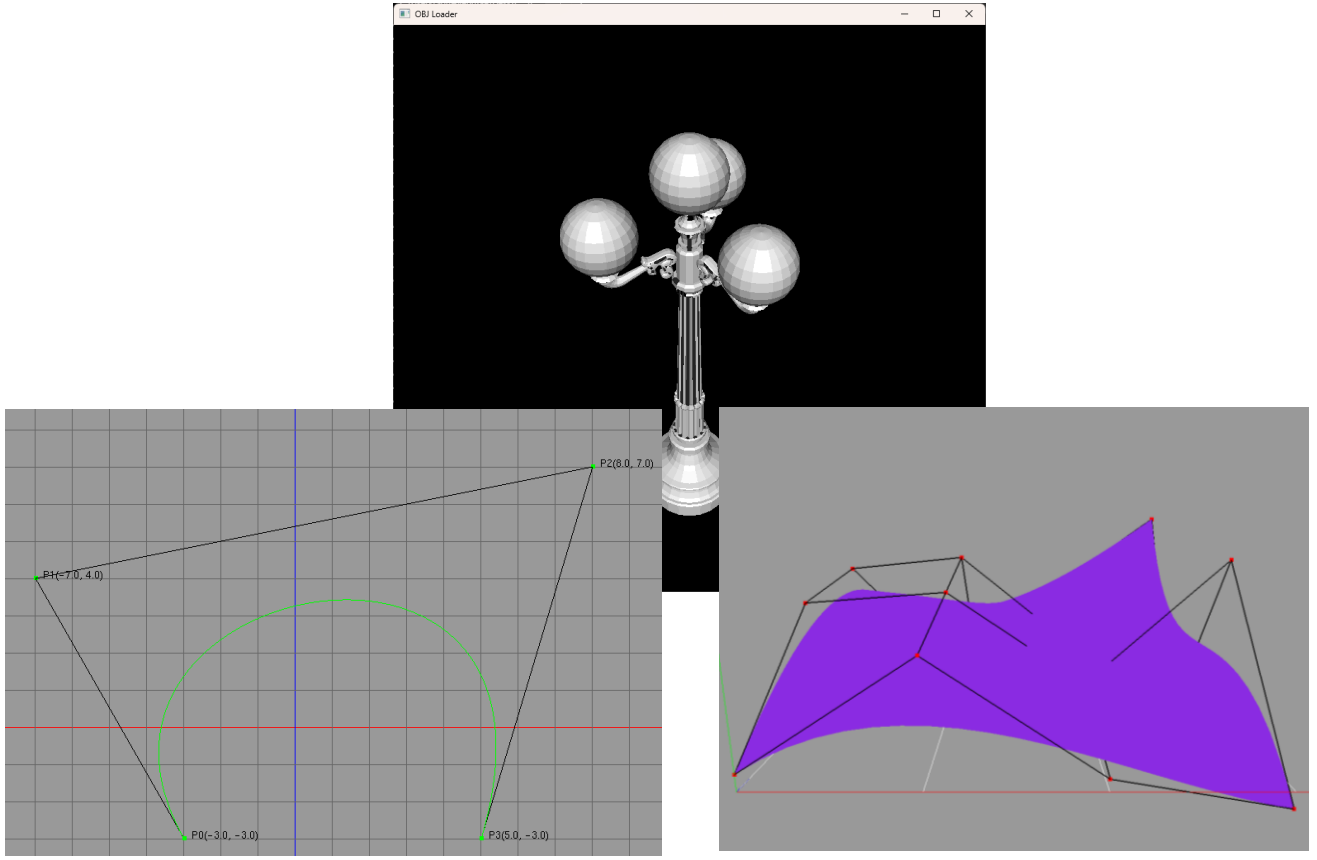
Computer Graphics Assignment #03

3D Object Modeling

과제 안내

- 과제 3 의 목표

1. 교재 8장 실습
    1. Polygon 모델링 실습
    2. 3차 베지어 곡선 구현
    3. 베지어 곡면 구현



< 과제 3 완료 결과 >

과제 안내

- .pdf 파일이 아닌 다른 파일로 제출하시는 분들이 많습니다.
- 문서작성 프로그램(.hwp, .word, etc.)를 사용하여

  과제 작성 후, .pdf로 저장하여 제출해주시기 바랍니다.

참고 자료

- Wavefront .obj file (Wikipedia)
  https://en.wikipedia.org/wiki/Wavefront_.obj_file#:~:text=OBJ%20(or%20.,OBJ%20geometry%20format

- Source of .obj Files
  https://people.sc.fsu.edu/~jburkardt/data/obj/obj.html
  (실습용 코드은 삼각, 사각 mesh만 지원하도록 코딩됨. 모든 obj 파일이 돌아가지는 않음.)

- 베지어 곡선 설명 (한글로 잘 설명된 페이지들)
  https://choi-dan-di.github.io/computer-graphics/curves-and-surfaces/#-%EB%B2%A0%EC%A7%80%EC%96%B4-%EC%82%BC%EA%B0%81%ED%98%95

  https://lee-seokhyun.gitbook.io/game-programming/client/easy-mathematics/gdc2012/1-2-3

- Source Bezier curve image(Korea University Media Lab.)
  https://media.korea.ac.kr/books/

과제 제출 방식

1. 코드의 모든 줄에는 주석이 포함되어야 함.
    1. int main(void); return 0; 등의 기초 라인들은 제외해도 무방
2. 추가 라이브러리 존재 시, 주석으로 설명
3. 실행한 코드와 실행 결과(capture)를 .pdf 파일로 제출
4. LMS 시스템에 제출

문제 1

Polygon Mesh practice

(textbook page 230)

문제 1

- Wavefront .obj 파일을 glut를 이용하여 화면에 띄우기
  (.obj 파일과 구현된 제공됨)

1. 제공된 코드 중, 핵심 부분 주석 작성

2. .obj 파일 중 2가지 혹은 그 이상을 실행,

   실행결과 캡쳐하여 제출

문제 1 – 배경 이론

Wavefront .obj

- Wavefront Technologies에서 개발된 표준 파일 형식으로 3차원 모델의 정점(Vertices), 면(Faces), 텍스처 좌표 등의 구조를 저장하는데 사용됨.

- Blender, 3D Max, AutoCAD, MeshLab 등의 S/W에서 이용

문제 1 – 배경 이론

Wavefront .obj 의 구조

icosahedron.obj

```
v  0  -0.525731  0.850651
v  0.850651  0  0.525731
v  0.850651  0  -0.525731
v  -0.850651  0  -0.525731
v  -0.850651  0  0.525731
v  -0.525731  0.850651  0
v  0.525731  0.850651  0
v  0.525731  -0.850651  0
v  -0.525731  -0.850651  0
v  0  -0.525731  -0.850651
v  0  0.525731  -0.850651
v  0  0.525731  0.850651

f  2  3  7
f  2  8  3
f  4  5  6
f  5  4  9
f  7  6  12
f  6  7  11
f  10  11  3
f  11  10  4
f  8  9  10
f  9  8  1
f  12  1  2
f  1  12  5
f  7  3  11
f  2  7  12
f  4  6  11
f  6  5  12
f  3  8  10
f  8  2  1
f  4  10  9
f  5  9  1
```

- vertices

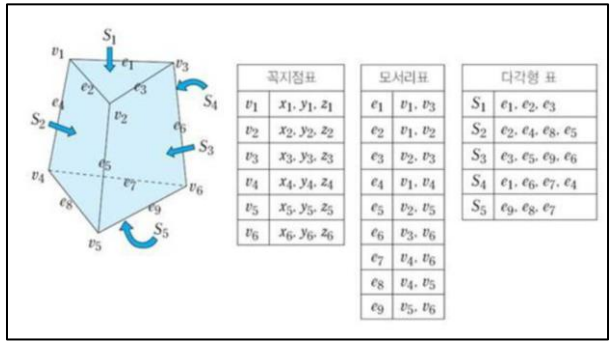
- faces

구현 필요

decode

3D object

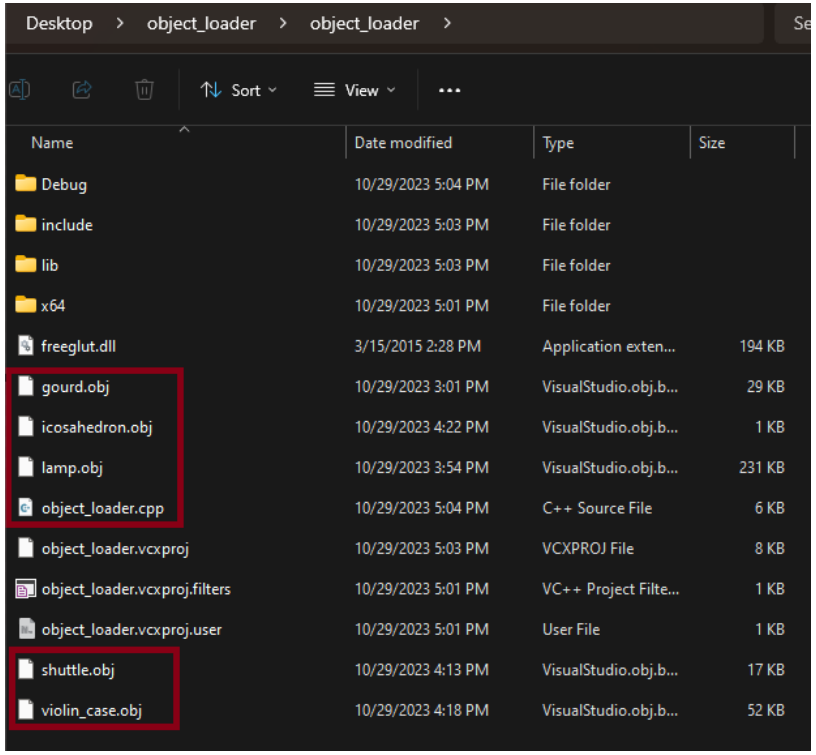
(practice decoding using GLUT)



참고

문제 1 – step 1

- 제공되는 파일을 프로젝트 폴더 내로 이동
  - object_loader.cpp
  - .obj files
    - gourd.obj
    - icosahedron.obj
    - lamp.obj
    - shuttle.obj
    - violin_case.obj



문제 1 – step 2

- Run object_loader.cpp

  : 확인하고자 하는 object의 주석 해제 후 실행

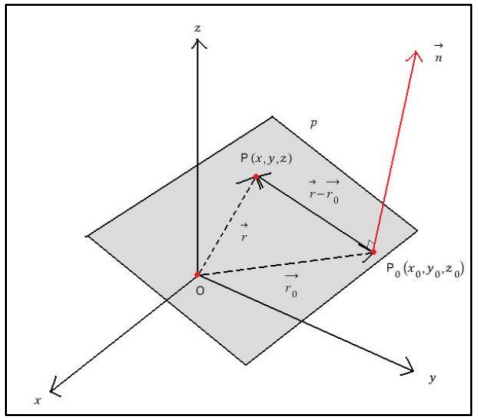
```
void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    gluLookAt(0, -3, 5, 0, 0, 0, 0, 0, 1); // gourd
    //gluLookAt(3, 3, 3, 0, 0, 0, 0, 0, 1); // icosahedron
    //gluLookAt(8, 15, 8, 0, 4, 0, 0, 1, 0); // lamp
    //gluLookAt(-10, 10, 15, 0, 0, 0, 0, 0, 1); // shuttle
    //gluLookAt(3, 3, 3, 0, 0, 0, 0, 0, 1); // violin_case
```

```
int main() {
    std::string objFilePath = "./gourd.obj";
    //std::string objFilePath = "./icosahedron.obj";
    //std::string objFilePath = "./lamp.obj";
    //std::string objFilePath = "./shuttle.obj";
    //std::string objFilePath = "./violin_case.obj";
```

문제 1 – step 3

- 구현 핵심부 주석 작성 (display 함수 내 위치)

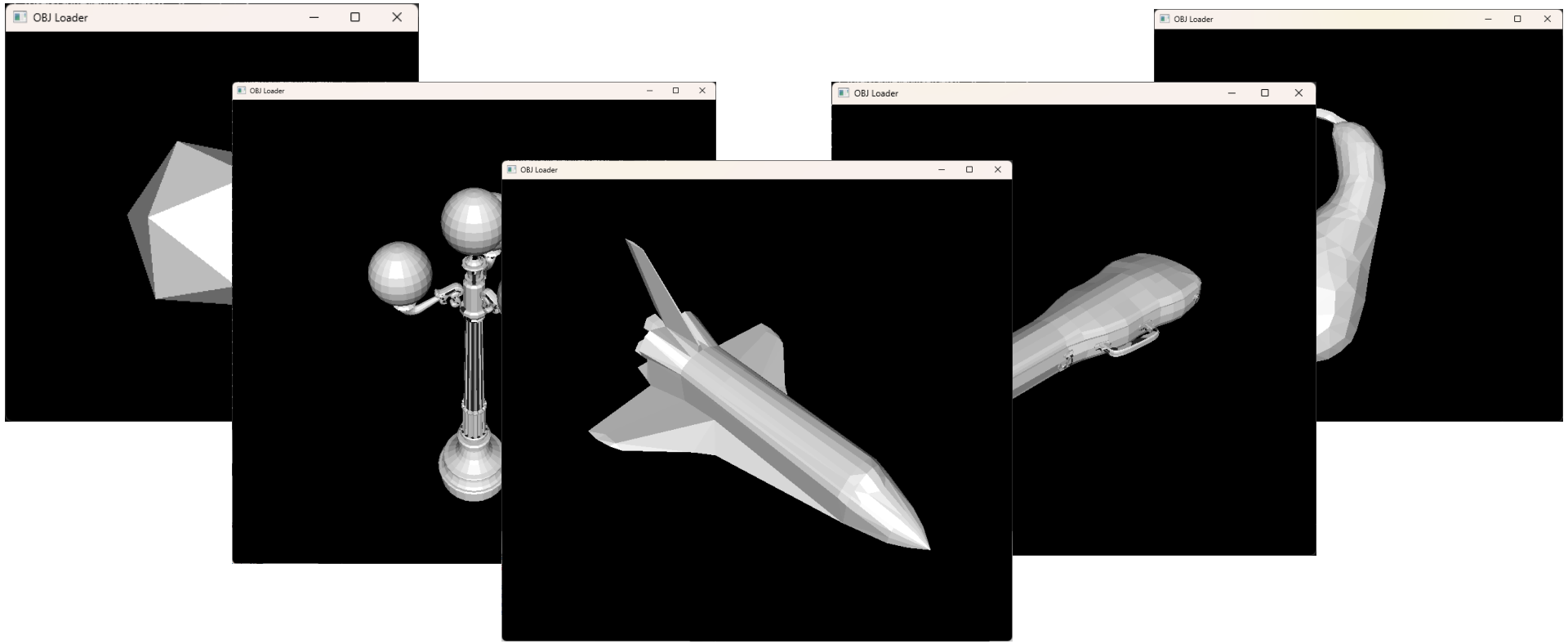
```
// 모든 줄 주석 작성 후 캡쳐
for (const auto& face : faces) {
    if (face.type == TRIANGLE) {
        glBegin(GL_TRIANGLES);
        if (face.vertices.size() >= 3) {
            Vertex& v1 = vertices[face.vertices[0]];
            Vertex& v2 = vertices[face.vertices[1]];
            Vertex& v3 = vertices[face.vertices[2]];

            Vertex vector1 = { v2.x - v1.x, v2.y - v1.y, v2.z - v1.z };
            Vertex vector2 = { v3.x - v1.x, v3.y - v1.y, v3.z - v1.z };
            Vertex normal = crossProduct(vector1, vector2);
            normalize(normal);
            glNormal3f(normal.x, normal.y, normal.z);
        }
        for (int vertexIndex : face.vertices) {
            glVertex3f(vertices[vertexIndex].x, vertices[vertexIndex].y, vertices[vertexIndex].z);
        }
        glEnd();
    }
    else if (face.type == SQUARE) {
        glBegin(GL_QUADS);
        if (face.vertices.size() >= 4) {
            Vertex& v1 = vertices[face.vertices[0]];
            Vertex& v2 = vertices[face.vertices[1]];
            Vertex& v3 = vertices[face.vertices[2]];
            Vertex& v4 = vertices[face.vertices[3]];

            Vertex vector1 = { v2.x - v1.x, v2.y - v1.y, v2.z - v1.z };
            Vertex vector2 = { v3.x - v1.x, v3.y - v1.y, v3.z - v1.z };
            Vertex normal = crossProduct(vector1, vector2);
            normalize(normal);
            glNormal3f(normal.x, normal.y, normal.z);
        }
        for (int vertexIndex : face.vertices) {
            glVertex3f(vertices[vertexIndex].x, vertices[vertexIndex].y, vertices[vertexIndex].z);
        }
        glEnd();
    }
}
```



참고

문제 1

- Outputs



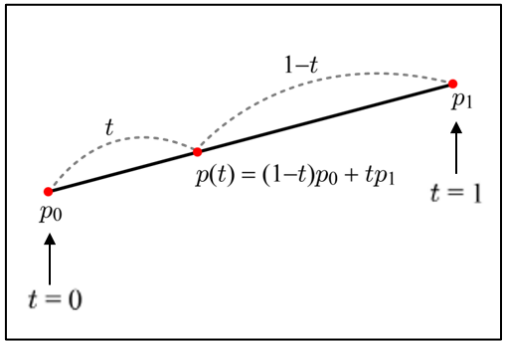
문제 2

cubic Bezier Curve implement

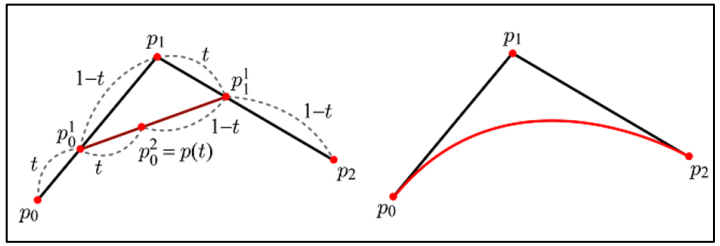
(textbook page 243)

문제 2

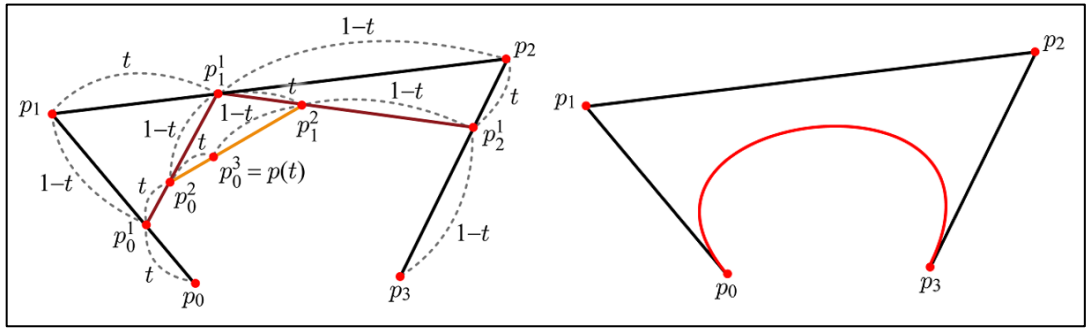
- 1차/2차 베지어 커브 이론을 바탕으로 3차 베지어 커브 구현



1차



2차



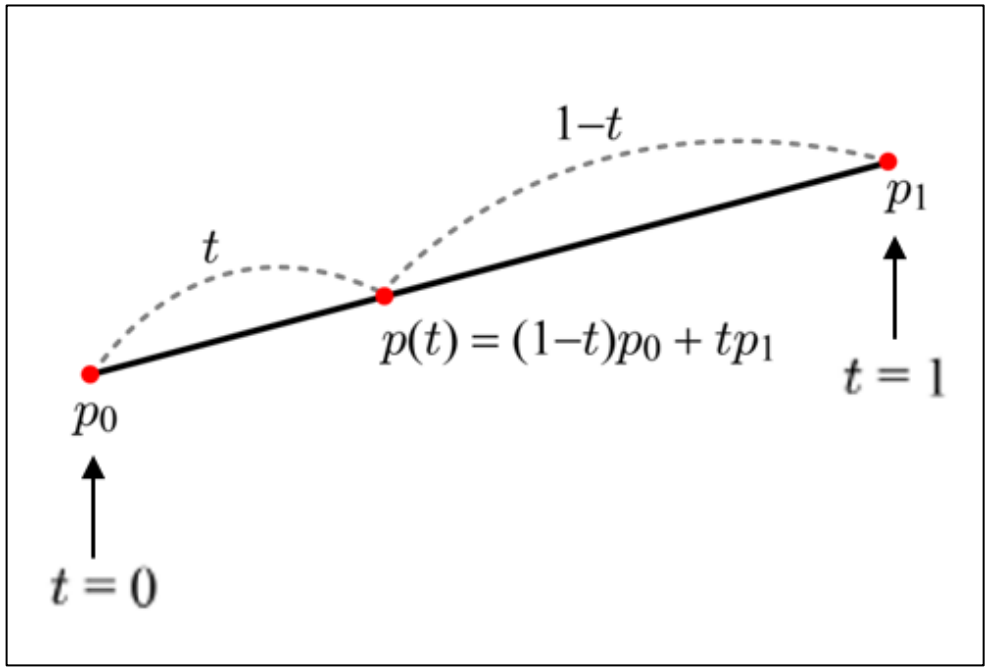
3차

문제 2 – 배경 이론

- 기본 이론 → 교재 p.243 참조
- 심화 이론 → 참고자료
- 본 과제에서는 spline curve 중, Bezier curve를 실습함.

문제 2 – 배경 이론

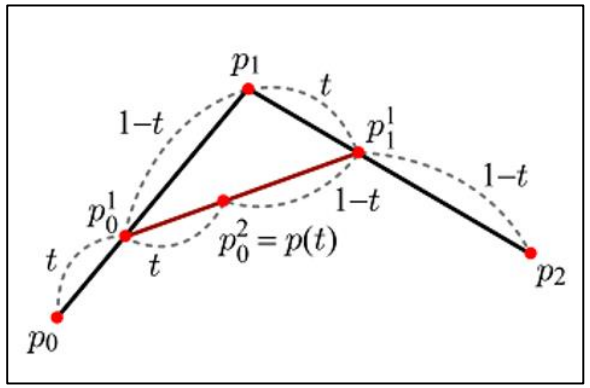
- 1차 베지어 == 선형 보간

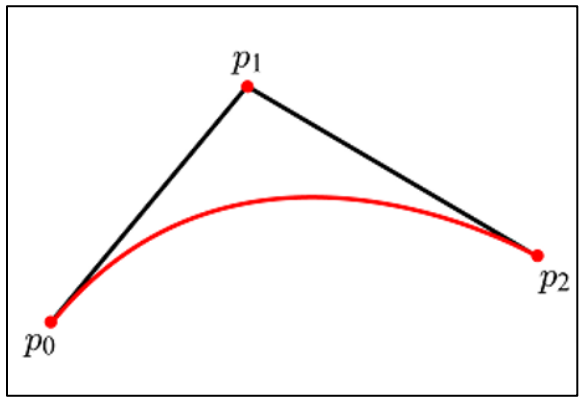


- p_0와 p_1사이를 보간하는 것이 목표.
- 선분은 점 두개를 요하지만,
  곡선의 보간을 위해선 3개 이상의 점이 필요.
  (결국 선분이 되는 것은 마찬가지)
- 점 두개의 보간 함수는 다음과 같음
  $p(t) = (1-t)p_0 + tp_1$

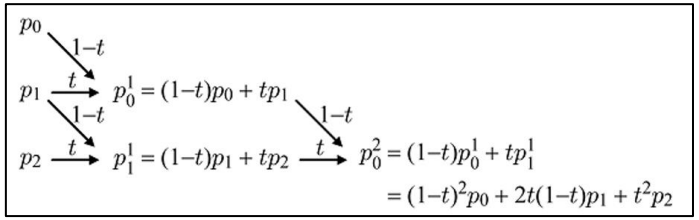
문제 2 – 배경 이론

- 2차 베지어 곡선



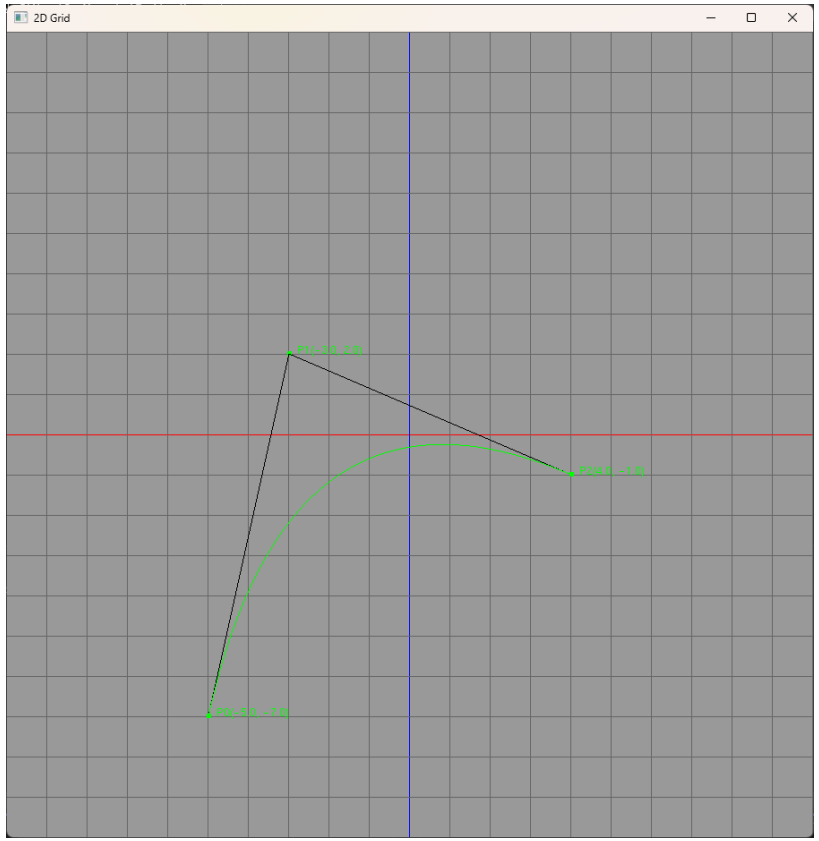


- P_0, P_1, P_2를 보간하는 것이 목표.
  이를 위해선 P_0-P_1, P_1-P_2 총 두번의 보간
- $p_0^1 = (1 - t)p_0 + tp_1$
  $p_1^1 = (1 - t)p_1 + tp_2$
- 두번의 보간을 다시 보간하면
  $p_0^2 = (1 - t)p_0^1 + tp_1^1$
- 위의 수식으로, 베지어 곡선은
  재귀적(recursion)임을 알 수 있음



문제 2 – 배경 이론

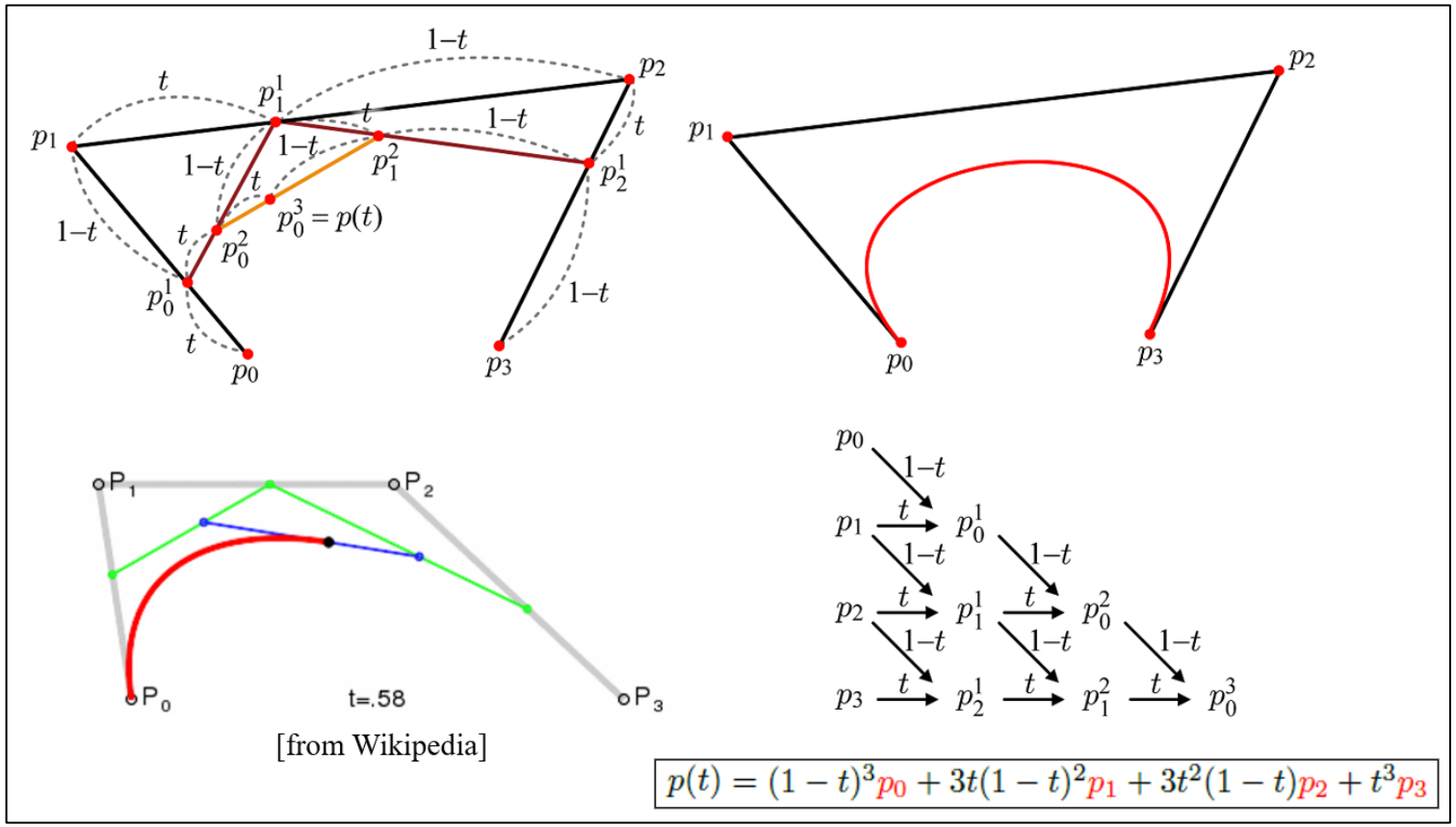
- 2차 베지어 곡선 → 구현된 코드가 제공됨.



bezier_curve.cpp

문제 2

- 2차 베지어 곡선 예시 코드를 참고하여 3차 베지어 곡선 구현

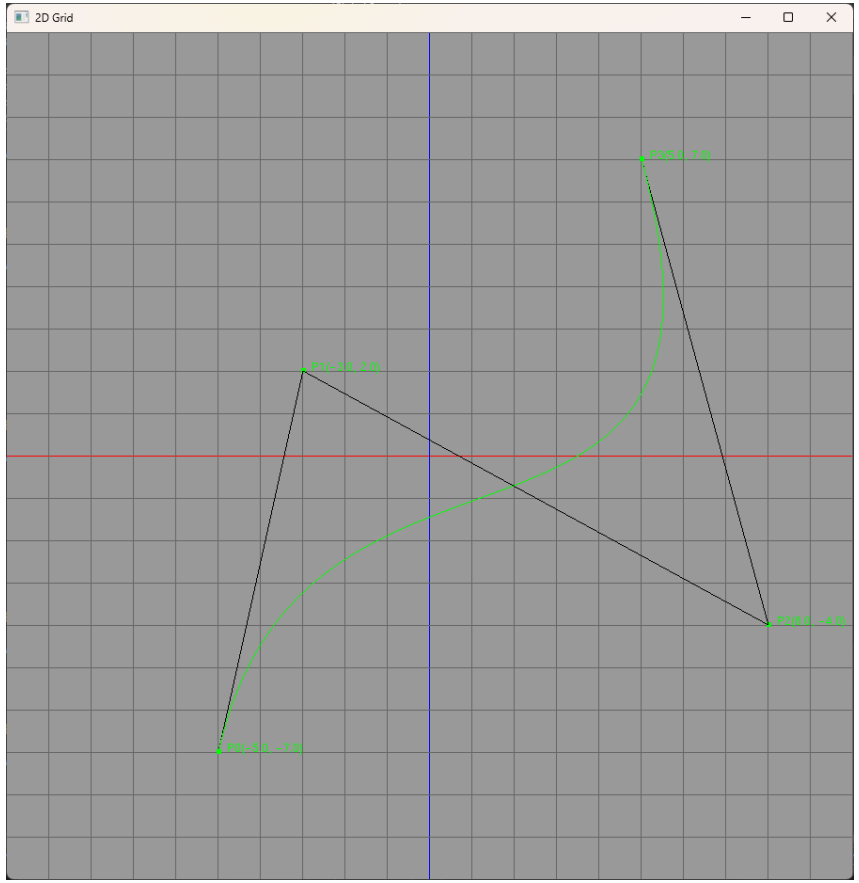


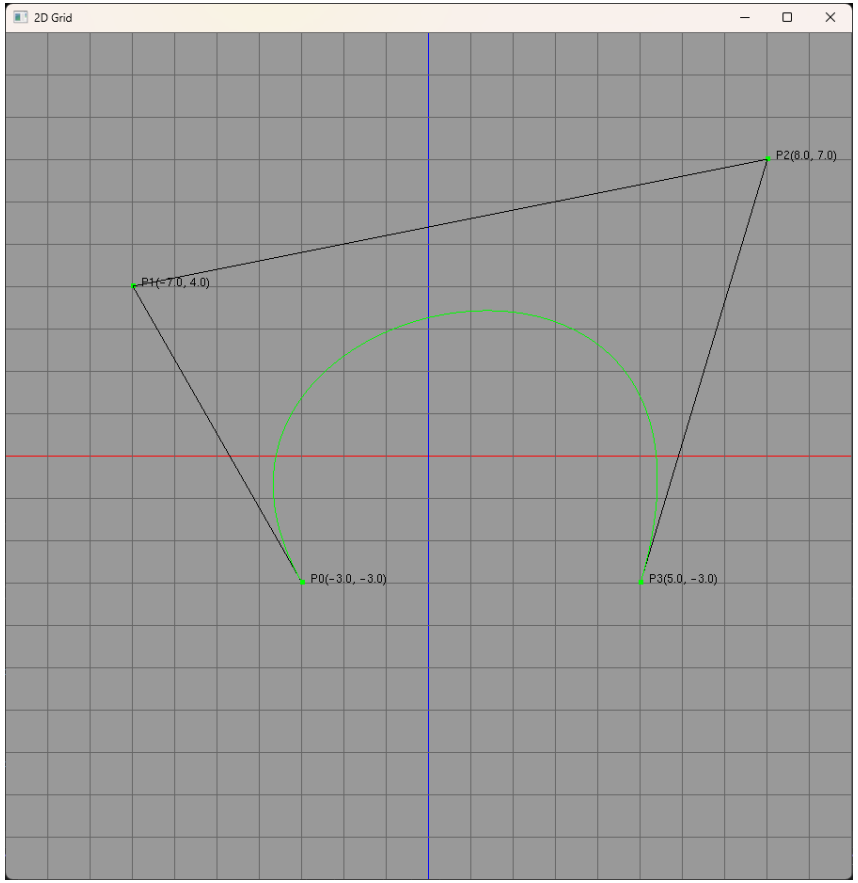
[from Wikipedia]

$$p(t) = (1-t)^3 p_0 + 3t(1-t)^2 p_1 + 3t^2(1-t)p_2 + t^3 p_3$$

문제 2 – 구현 결과

- 2차 베지어 곡선 예시 코드를 참고하여 3차 베지어 곡선 구현





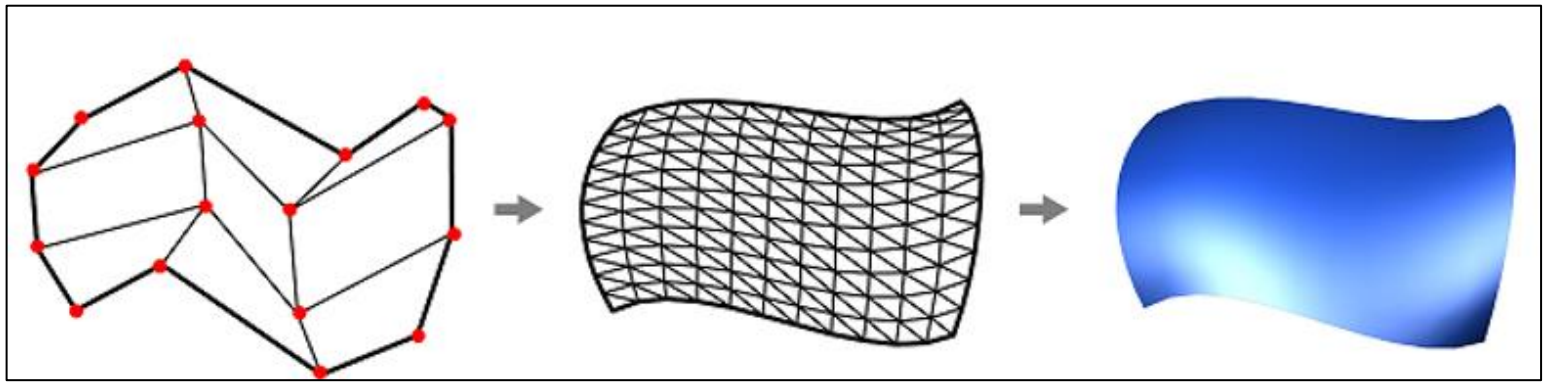
문제 3

Bezier Surface implement

(textbook page 246)

문제 3

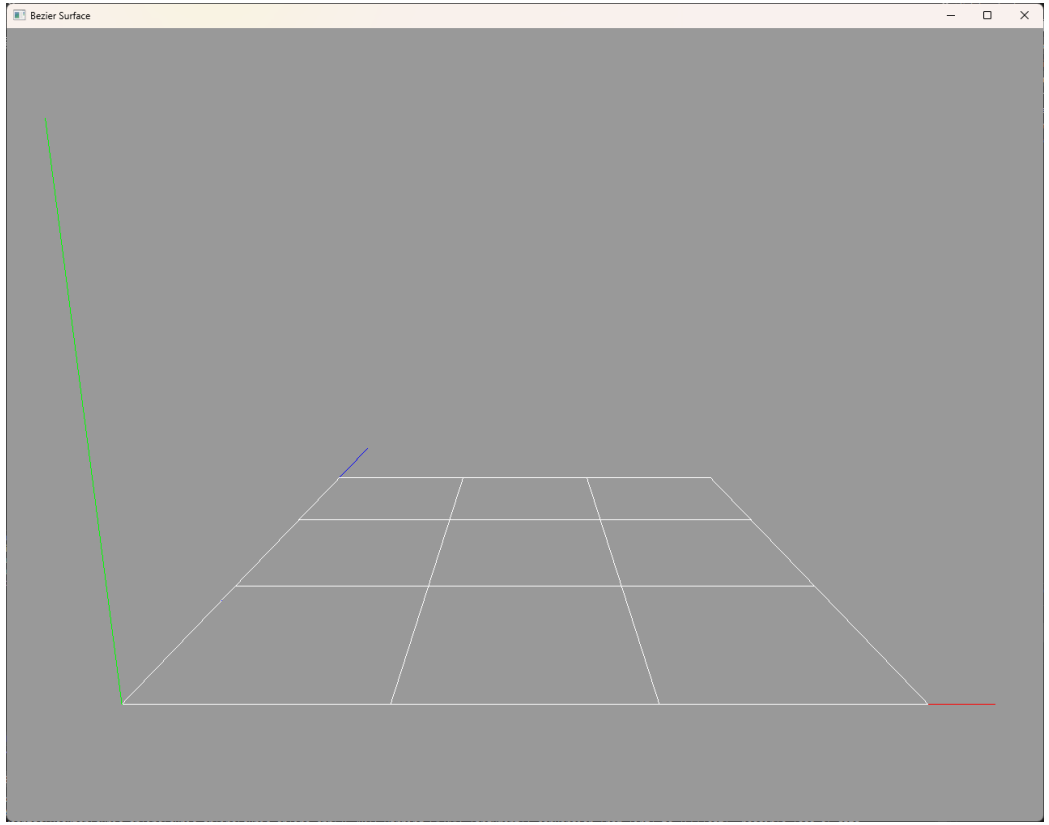
- 3차 베지어 커브 구현을 바탕으로 3차 베지어 곡면 구현



구현 핵심 수식 : $P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} P_{j,k} B_{j,m}(v) B_{k,n}(u)$

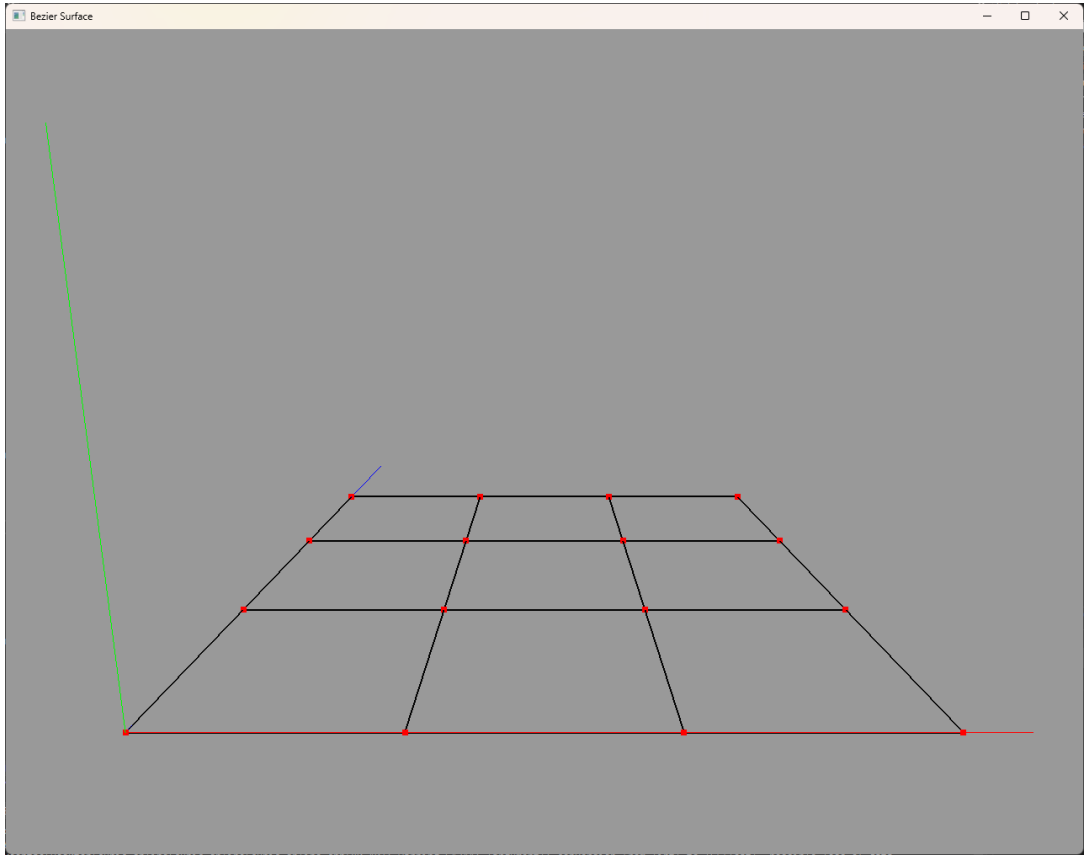
문제 3 – step 1

- <과제 1>의 5번 문항 코드를 base로 하여 시작



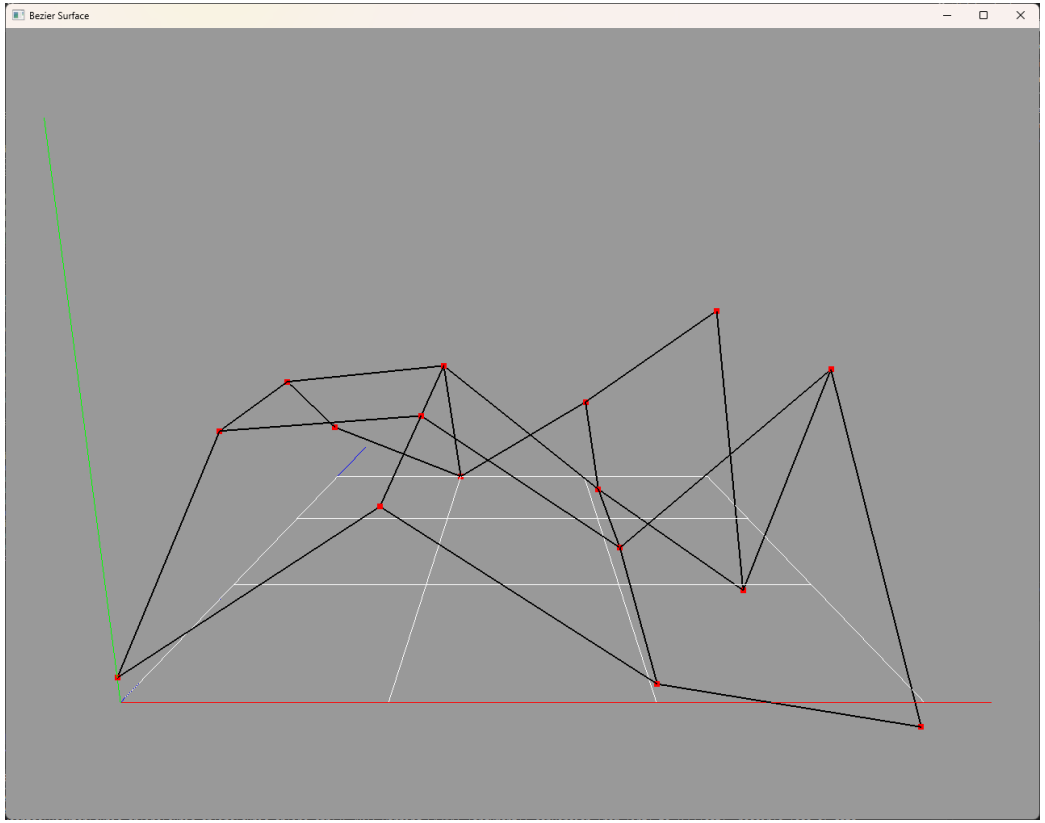
문제 3 – step 2

- Control point 생성 후 상호 연결 (해당 부분까지 코드 제공)



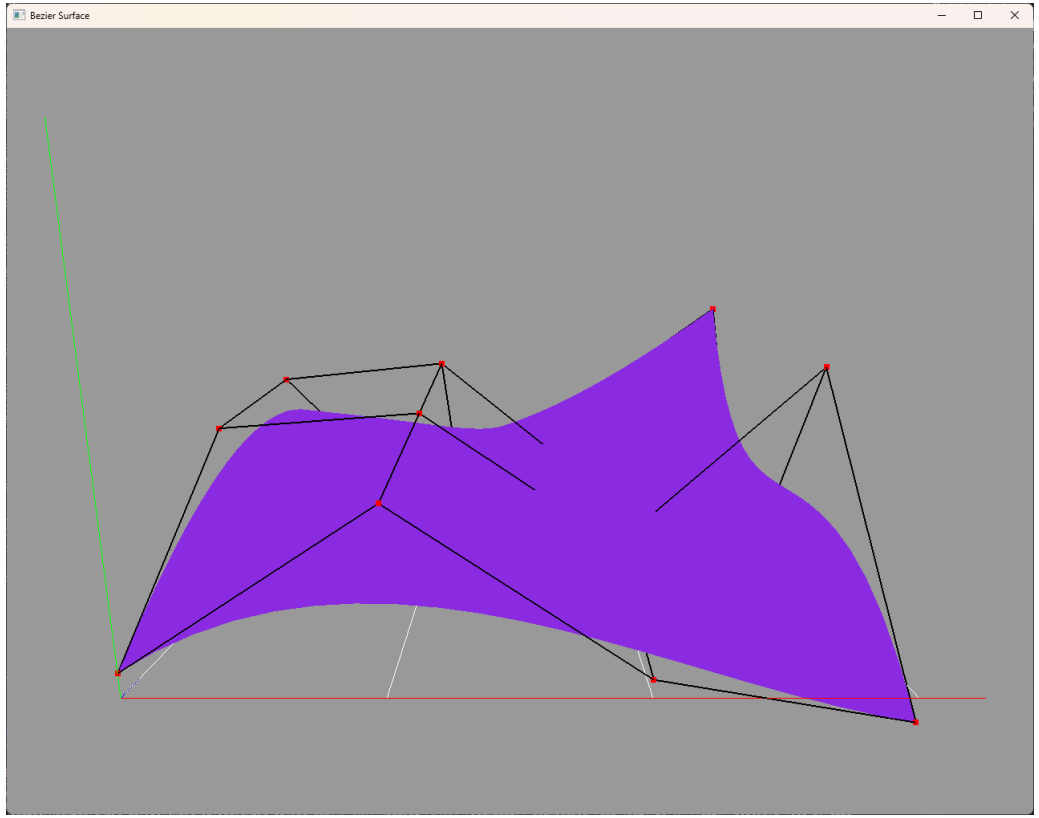
문제 3 – step 3

- Control point 이동



문제 3 – step 4

- Bezier Surface 구현



문의 (연구생 손현태)

- 1. Mail : lmstk @ kyonggi . ac . Kr
- 2. Tel : 010 – 4223 – 4517
- 3. 컴퓨터 그래픽스 & 이미지처리 연구실 (8501호) 방문