

(기말고사 대체 프로젝트)

컴퓨터 그래픽스

- SOR Modeling data를 이용한 가상 공간 렌더링 -



중앙대학교 예술공학대학 컴퓨터예술학부

20194004

양소영

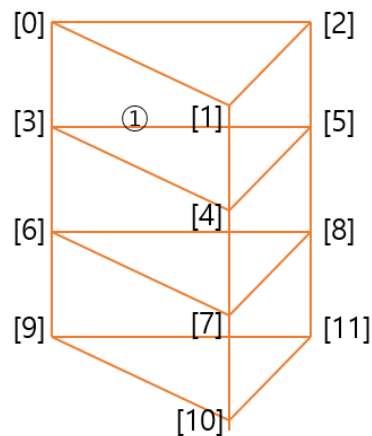
1. 문제 내용 및 해결 방안

본격적으로 스크립트를 짜기에 앞서, 다음 순으로 단계를 나누어 차근차근 구현해보기로 하였다.

중간고사 코드 수정 > SOR 모델링 데이터 저장 > 바닥 추가 > BackFace Culling 기능 추가 > Smooth shading 구현 > 투영방식 변환 > 카메라 회전 및 시점 변환 > 기타 기능 구현

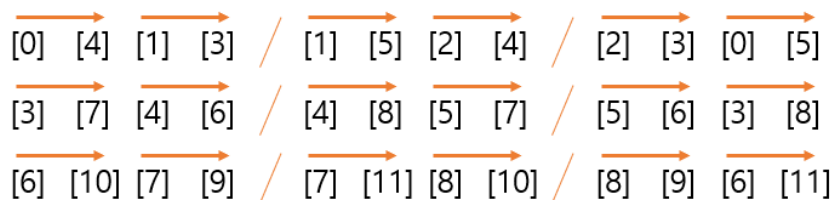
(1) 중간고사 코드 수정

기존 코드에서는 삼각형의 wireframe 을 그리기 위해 cross 라는 함수를 정의하여 모델 예시 [그림 1]의 각 면에 사선을 그려주었다.



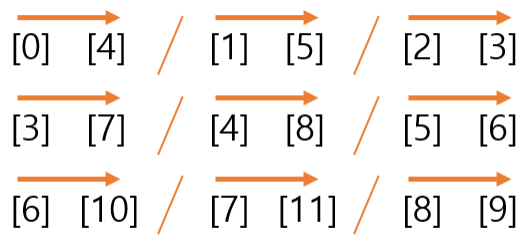
[그림 1]

cross 함수는 ①과 같은 면 마다 'X'자로 선을 그려주었는데, [그림 2]에서 보이는 것처럼 line 을 각각 그려주었다.



[그림 2]

하지만 SOR 모델링 데이터를 저장하기 위해서는 한 면 당 사선이 한 개만 필요했으며, [그림 3]과 같이 좌하향 사선을 지워야 했다. 따라서 cross 함수를 수정하는 과정을 거쳤다.



[그림 3]

```
void cross(int c) {
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_LINE_STRIP); // wireframe 삼각형 그리기
    for (int i = 0; i < (sweepResolutionMod - 1); i++) {
        glVertex3f(arRotPoints[c * sweepResolutionMod + i].x, arRotPoints[c * sweepResolutionMod + i].y, arRotPoints[c * sweepResolutionMod + i].z);
        glVertex3f(arRotPoints[c * sweepResolutionMod + (sweepResolutionMod + 1) + i].x, arRotPoints[c * sweepResolutionMod + (sweepResolutionMod + 1) + i].y,
            arRotPoints[c * sweepResolutionMod + (sweepResolutionMod + 1) + i].z);
    }
    glVertex3f(arRotPoints[c * sweepResolutionMod + (sweepResolutionMod - 1)].x, arRotPoints[c * sweepResolutionMod + (sweepResolutionMod - 1)].y,
        arRotPoints[c * sweepResolutionMod + (sweepResolutionMod - 1)].z);
    glVertex3f(arRotPoints[c * sweepResolutionMod + (sweepResolutionMod)].x, arRotPoints[c * sweepResolutionMod + sweepResolutionMod].y,
        arRotPoints[c * sweepResolutionMod + sweepResolutionMod].z);
    glVertex3f(arRotPoints[c * sweepResolutionMod].x, arRotPoints[c * sweepResolutionMod].y, arRotPoints[c * sweepResolutionMod].z);

    glEnd();
    glFlush();
}
```

(2) SOR 모델링 데이터 저장

모델링 데이터를 저장하기 위해서는 저장하는 함수가 따로 필요하였다.

```
void SaveModel() {
    FILE* fout;

    fout = fopen("c:\\data\\myModel.dat", "w");

    fprintf(fout, "VERTEX = %d\n", pnum);
    for (int i = 0; i < pnum; i++) {
        fprintf(fout, "%.1f %.1f %.1f\n", mpoint[i].x, mpoint[i].y, mpoint[i].z);
    }

    fprintf(fout, "FACE = %d\n", fnum);
    for (int i = 0; i < fnum; i++) {
        fprintf(fout, "%d %d %d\n", mface[i].ip[0], mface[i].ip[1], mface[i].ip[2]);
    }
    fclose(fout);
}
```

교안을 참고하여 SaveModel 이라는 함수를 정의해주었고, 여기서 mpoint, mface 라는 배열을 만들어 해당하는 값을 각각 삽입할 필요가 있었다.

```

pnum = arRotPoints.size(); // point 개수 지정
fnum = sweepResolutionMod * (arInputPoints.size() - 1) * 2; // face 개수 지정
mpoint = new Point[pnum];
mface = new Face[fnum];

for (int i = 0; i < pnum; i++) {
    mpoint[i].x = arRotPoints[i].x * 0.7;
    mpoint[i].y = arRotPoints[i].y * 0.7;
    mpoint[i].z = arRotPoints[i].z * 0.7;
}

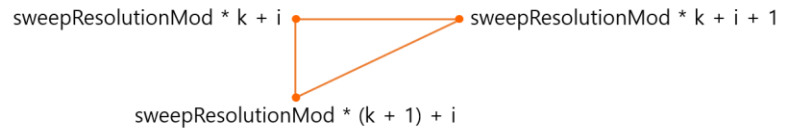
for (int k = 0; k < arInputPoints.size() - 1; k++) {
    for (int i = 0; i < sweepResolutionMod - 1; i++) { // up-triangle
        mface[cnum].ip[1] = sweepResolutionMod * k + i + 1;
        mface[cnum].ip[2] = sweepResolutionMod * k + i;
        mface[cnum].ip[0] = sweepResolutionMod * (k + 1) + i;
        cnum += 1;
    }
    mface[cnum].ip[1] = sweepResolutionMod * k;
    mface[cnum].ip[2] = sweepResolutionMod * (k + 1) - 1;
    mface[cnum].ip[0] = sweepResolutionMod * (k + 2) - 1;
    cnum += 1;

    for (int i = 0; i < sweepResolutionMod - 1; i++) { // down-triangle
        mface[cnum].ip[0] = sweepResolutionMod * k + i + 1;
        mface[cnum].ip[1] = sweepResolutionMod * (k + 1) + i;
        mface[cnum].ip[2] = sweepResolutionMod * (k + 1) + i + 1;
        cnum += 1;
    }
    mface[cnum].ip[0] = sweepResolutionMod * (k);
    mface[cnum].ip[1] = sweepResolutionMod * (k + 2) - 1;
    mface[cnum].ip[2] = sweepResolutionMod * (k + 1);
    cnum += 1;
}
SaveModel();
}

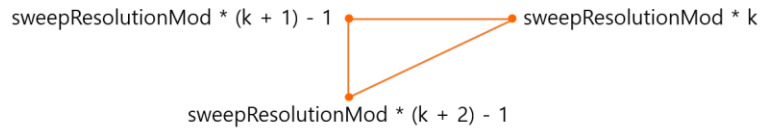
```

pnum 은 point 배열의 개수로 지정해야 하기 때문에 입력 받은 모든 점의 개수로 정의하였고, fnum 은 face 배열의 개수로 지정해야 하기 때문에 모델을 이루고 있는 사각형 면의 개수에 2 배를 하여 정의해주었다. 이렇게 각 배열을 선언해준 후, mpoint 배열에는 입력 받은 모든 점 값을 담고 있는 arRotPoints 의 배열을 넣어주었으며 mface 배열에는 삼각형 면을 이루고 있는 mpoint 의 순서를 3 개씩 담아야 했는데 이 알고리즘을 생각하는 과정이 꽤 까다로웠다. 나의 경우는 각 층을 이루고 있는 삼각형 모양의 면을 [그림 4], [그림 5]와 같이 위 삼각형과 아래 삼각형으로 나누어 생각해보았다.

k층에 생기는 삼각형



k층에 생기는 마지막 삼각형

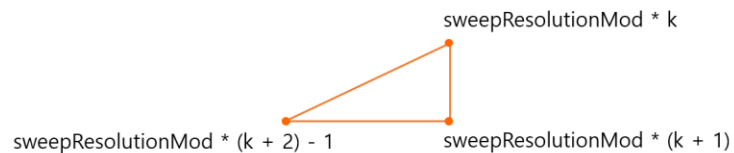


[그림 4] 위 삼각형

k층에 생기는 삼각형



k층에 생기는 마지막 삼각형



[그림 5] 아래 삼각형

각 층마다 삼각형을 이루는 각 point 가 반복되지만, 층의 마지막 삼각형은 처음 삼각형과 이어져야 하기 때문에 반복문에 포함시키지 않고 따로 입력해주어야 한다. 이렇게 코드를 구현하고 프로그램을 실행시키면 점을 찍는 화면이 나타나고, 각도를 입력하여 회전시키면 바로 c:\wwdata 폴더에 myModel.dat 파일이 저장된다.

(3) 바닥 추가

바닥 추가를 구현하는 것은 8 번째 교안 2 번 예시를 참고하여 코드를 작성하였다. 가로, 세로는 (400, 400) 길이가 되도록 만들어주었고 DrawWire 과 DrawShade 함수에 각각 넣어주었다.

```

glColor3f(0.7, 0.7, 0.7);
glPushMatrix();
glTranslatef(0, -100, 0);
glBegin(GL_QUADS);
glVertex3f(200, 0, 200);
glVertex3f(200, 0, -200);
glVertex3f(-200, 0, -200);
glVertex3f(-200, 0, 200);
glEnd();
glPopMatrix();
glutSwapBuffers();

```

(4) BackFace Culling 기능 추가

처음에 모델링 데이터를 불러오면 wireframe 모드로 보여지는데, 이 때의 모델링은 뒷면까지 보이는 상태이다. 여기서 glEnable 함수를 사용하면 제거할 부분의 면을 선택할 수 있다. glEnable(GL_CULL_FACE) 함수를 불러 면 제거를 가능하도록 변경해주고, glCullFace(GL_BACK)을 넣어주면 모델의 뒷면을 제거할 수 있다. 키보드 'e'를 눌렀을 때 BackFace Culling 기능을 On 시킬 수 있다. 만약 이 기능을 다시 Off 시켜주고 싶다면 키보드 'd'를 누르면 된다. 구글링을 하다가 우연히 glDisable 이라는 함수를 발견하게 되었는데, 뒷면을 제거한 상태에서 GL_CULL_FACE 기능을 가능하지 못하게 하면 제거되었던 뒷면이 다시 보이게 된다.

```

case 'e': // hide back
    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);
    glutPostRedisplay();
    break;
case 'd': // show back
    glDisable(GL_CULL_FACE);
    glutPostRedisplay();
    break;

```

(5) Smooth shading 구현

조명을 비출 때 빛이 모델링의 각 면에서 반사되어 부드럽게 보이려면 한 vertex 를 기준으로 그 점에 인접해 있는 모든 면의 법선 벡터를 더해서 평균을 구한 뒤, 조명이 해당 평균 벡터로부터 빛을 반사해야 한다. 처음에는 Gouraud shading 을 구현하기 위해 다음과 같이 코드를 작성하여 평균 벡터를 모두 구하려는 시도를 하였다.

```

Point newnormal(Point a) { // smooth shading
    Point n;

    n.x = 0;
    n.y = 0;
    n.z = 0;

    for (int i = 0; i < pnum; i++) {
        int n_count = 0;
        for (int j = 0; j < fnum; j++) {
            if (i == mface[j].ip[0] || i == mface[j].ip[1] || i == mface[j].ip[2]) {
                a.x += a.x;
                a.y += a.y;
                a.z += a.z;
                n_count++;
            }
        }
        n.x = a.x / n_count;
        n.y = a.y / n_count;
        n.z = a.z / n_count;
    }
    return n;
}

```

하지만 작동이 잘 되지 않아 굉장히 많은 시간을 지체하는 시행착오를 겪었으며, 구글링을 하다가 glNormal3f 라는 함수가 vertex에 대한 법선 벡터를 정의하는 함수라는 것을 알게 되어 이 함수의 정확한 쓰임새를 알게 되었다.

```

glEnable(GL_NORMALIZE);
for (i = 0; i < fnum; i++) {
    Point norm = mpoint[mface[i].ip[0]];
    glBegin(GL_TRIANGLES);

    glNormal3f(norm.x, norm.y, norm.z); // 면을 이루는 첫번째 vertex
    glVertex3f(mpoint[mface[i].ip[0]].x, mpoint[mface[i].ip[0]].y, mpoint[mface[i].ip[0]].z);

    norm = mpoint[mface[i].ip[1]];
    glNormal3f(norm.x, norm.y, norm.z); // 면을 이루는 두번째 vertex
    glVertex3f(mpoint[mface[i].ip[1]].x, mpoint[mface[i].ip[1]].y, mpoint[mface[i].ip[1]].z);

    norm = mpoint[mface[i].ip[2]];
    glNormal3f(norm.x, norm.y, norm.z); // 면을 이루는 세번째 vertex
    glVertex3f(mpoint[mface[i].ip[2]].x, mpoint[mface[i].ip[2]].y, mpoint[mface[i].ip[2]].z);
    glEnd();
}
glPopMatrix();
glEndList();

```

먼저 glEnable(GL_NORMALIZE)을 정의하여 단위 법선 벡터를 계산할 수 있도록 해주고, for 반복문 바로 다음에 norm 을 한 번만 설정해준 전과는 달리, 면을 이루는 vertex 전마다 각각 norm 을 적용시켜 주었더니 각 vertex 마다 법선 벡터를 지정해주어 빛이 들어오는 방향과 각을 다르게 계산하여 최종적으로 부드러운 느낌의 표면을 구현할 수 있게 되었다.

(6) 투영방식 변환

원근 투영과 직교 투영을 구현해주기 위해 `perspmode`, `orthomode` 함수를 정의해주어 투영 방식을 따로 지정해주었다.

```
void perspmode() { // perspective mode
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(40.0, 1.0, 1.0, 2000.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(600.0, 600.0, 600.0, 0.0, 0.0, 0.0, 1.0, 0.);
}

void orthomode() { // ortho mode
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-100.0, 100.0, -100.0, 100.0, -800.0, 800.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(600.0, 600.0, 600.0, 0.0, 0.0, 0.0, 1.0, 0.);
}
```

처음에 모델을 불러올 때는 원근 투영이 되어 있는 상태이다. 이 때 키보드 'f'버튼을 누르면 `orthomode` 함수를 불러와 모델이 직교 투영되어 있는 것을 볼 수 있고, 키보드 'r'버튼을 누르면 `perspmode` 함수를 불러와 모델이 다시 원근 투영되어 있는 것을 볼 수 있다.

```
case 'r': // perspective mode
    display();
    perspmode();
    glutPostRedisplay();
    break;
case 'f': // ortho mode
    display();
    orthomode();
    glutPostRedisplay();
    break;
```

(7) 카메라 회전 및 시점 변환

모델을 여러 방향에서 보기 위해 `display` 함수 안에 있는 `gluLookAt` 함수의 카메라 위치를 바꾸면 카메라를 회전시킬 수 있었다. 먼저 `xmove`, `ymove`, `zmove` 이라는 전역변수를 선언하고 다음과 같이 키보드를 누를 때마다 원하는 값만큼 증가 혹은 감소시켜 카메라를 이동하도록 하였다. 키보드 버튼을 이용하여 공간을 이동하는 네비게이션 모드를 구현할 수 있었다.


```

case 'b': // xmoving
    xmove += 10;
    display();
    glutPostRedisplay();
    break;
case 'n': // ymoving
    ymove += 10;
    display();
    glutPostRedisplay();
    break;
case 'm': // zmoving
    zmove += 10;
    display();
    glutPostRedisplay();
    break;
case 'h': // xmoving
    xmove -= 10;
    display();
    glutPostRedisplay();
    break;
case 'j': // ymoving
    ymove -= 10;
    display();
    glutPostRedisplay();
    break;
case 'k': // zmoving
    zmove -= 10;
    display();
    glutPostRedisplay();
    break;

```

또한 물체를 바라보는 시점을 x 축, y 축, z 축마다 변경 가능하도록 카메라 위치를 바꿔주었다.

```

case 'q': // x viewport
    xmove = 600;
    ymove = 0;
    zmove = 0;
    down = 0;
    display();
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glutPostRedisplay();
    break;
case 'a': // y viewports
    xmove = 0;
    ymove = 600;
    zmove = 0;
    down = -1;
    display();
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glutPostRedisplay();
    break;
case 'z': // z viewport
    xmove = 0;
    ymove = 0;
    zmove = 600;
    down = 0;
    display();
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glutPostRedisplay();
    break;

```

(8) 기타 기능 구현

모델의 심미적 부분을 향상시키기 위해 색상 변경을 해주고자 하였다. 먼저 colorR, colorG, colorB 라는 전역변수를 선언하고 초기 모델은 검정색으로 지정한 후, 각 키보드 버튼을 누를 때마다 빨간색, 초록색, 파란색으로 변경되도록 해주었다.

```
case '1': // red color
    colorR = 0.8; colorG = 0.4; colorB = 0.4;
    display();
    glutPostRedisplay();
    break;
case '2': // green color
    colorR = 0.4; colorG = 0.8; colorB = 0.4;
    display();
    glutPostRedisplay();
    break;
case '3': // blue color
    colorR = 0.4; colorG = 0.4; colorB = 0.8;
    display();
    glutPostRedisplay();
    break;
```

2. 소스코드

다음은 SOR Modeling data 저장을 구현한 총 소스코드이다. 추후 과제 제출을 위해 상대방으로 저장하도록 스크립트를 수정하였다.

```
1  #define _USE_MATH_DEFINES
2  #define _CRT_SECURE_NO_WARNINGS
3
4  #include <GL/glut.h>
5  #include <stdlib.h>
6  #include <stdio.h>
7  #include <math.h>
8  #include <vector>
9  #include <iostream>
10 using namespace std;
11
12 #define WIRE 0
13 #define SHADE 1
14
15 #typedef struct {
16     float x;
17     float y;
18     float z;
19 } Point;
20
21 #typedef struct {
22     unsigned int ip[3];
23 } Face;
24
25 GLsizei winWidth = 1000, winHeight = 700;
26 GLboolean revolveFcn = false; // 점 회전
27 GLboolean clean = false; // 화면 초기화
28 GLboolean draw = true; // 점 찍기
29 GLboolean Wireframe = false; // wireframe 모드
30 GLfloat angle = 0;
31
32 int moving;
33 int mousebegin;
34 int light_moving;
35 float scaleFactor = 1.0;
36 int scaling = 0;
37 int status = 0;
38
39 float fRotAngle; // 입력받은 각도
40 float radian;
41 int sweepResolutionMod = (360 / fRotAngle); // 점이 회전하는 횟수
42
43
44 #class xPoint3D {
45 public:
46     float x, y, z, w;
47     xPoint3D() { x = y = z = 0; w = 1; };
48 };
49
50 xPoint3D pt;
51 vector<xPoint3D> arInputPoints; // 초기에 찍은 점 받기
52 vector<xPoint3D> arRotPoints; // 회전한 점 받기
53
54 int pnum;
55 int fnum;
56 Point* mpoint = NULL;
57 Face* mface = NULL;
58 int cnum = 0; // count number
59
60 void line(int l) {
61     glColor3f(1.0, 1.0, 0.0);
62     glBegin(GL_LINE_STRIP); // wireframe 행 그리기
63     for (int i = 0; i < sweepResolutionMod; i++) {
64         glVertex3f(arRotPoints[l + i].x, arRotPoints[l + i].y, arRotPoints[l + i].z);
65     }
66     glEnd();
67     glFlush();
68 }
69
70 void cross(int c) {
71     glColor3f(1.0, 1.0, 0.0);
72     glBegin(GL_LINE_STRIP); // wireframe 삼각형 그리기
73     for (int i = 0; i < (sweepResolutionMod - 1); i++) {
74         glVertex3f(arRotPoints[c * sweepResolutionMod + i].x, arRotPoints[c * sweepResolutionMod + i].y, arRotPoints[c * sweepResolutionMod + i].z);
```

```

75         glVertex3f(arRotPoints[c * sweepResolutionMod + (sweepResolutionMod + 1) + i].x, arRotPoints[c * sweepResolutionMod + (sweepResolutionMod + 1) + i].y,
76                   arRotPoints[c * sweepResolutionMod + (sweepResolutionMod + 1) + i].z);
77     }
78     glVertex3f(arRotPoints[c * sweepResolutionMod + (sweepResolutionMod - 1)].x, arRotPoints[c * sweepResolutionMod + (sweepResolutionMod - 1)].y,
79               arRotPoints[c * sweepResolutionMod + (sweepResolutionMod - 1)].z);
80     glVertex3f(arRotPoints[c * sweepResolutionMod + sweepResolutionMod].x, arRotPoints[c * sweepResolutionMod + sweepResolutionMod].y,
81               arRotPoints[c * sweepResolutionMod + sweepResolutionMod].z);
82     glVertex3f(arRotPoints[c * sweepResolutionMod].x, arRotPoints[c * sweepResolutionMod].y, arRotPoints[c * sweepResolutionMod].z);
83
84     glEnd();
85     glFlush();
86 }
87
88 void SaveModel() {
89
90     FILE* fout;
91
92     fout = fopen("c:\\data\\myModel.dat", "w");
93
94     fprintf(fout, "VERTEX = %d\n", pnum);
95     for (int i = 0; i < pnum; i++) {
96         fprintf(fout, "%.1f %.1f %.1f\n", mpoint[i].x, mpoint[i].y, mpoint[i].z);
97     }
98
99     fprintf(fout, "FACE = %d\n", fnum);
100    for (int i = 0; i < fnum; i++) {
101        fprintf(fout, "%d %d %d\n", mface[i].ip[0], mface[i].ip[1], mface[i].ip[2]);
102    }
103    fclose(fout);
104 }
105
106 void MyDisplay() {
107     glViewport(0, 0, 1000, 700);
108     glColor3f(1.0, 0.0, 0.0);
109
110     glColor3f(0.5, 0.5, 0.5); // 축 그리기
111     glBegin(GL_LINES);
112     glVertex3f(500.0, 0.0, 0.0); // x축
113     glVertex3f(-500.0, 0.0, 0.0);
114     glVertex3f(0.0, -500.0, 0.0); // y축
115     glVertex3f(0.0, 500.0, 0.0);
116     glVertex3f(0.0, 0.0, -500.0); // z축
117     glVertex3f(0.0, 0.0, 500.0);
118     glEnd();
119
120     if (revolveFcn) { // 점 회전
121
122         float fNewAngle = 0;
123         radian = fNewAngle * (M_PI / 180.0);
124
125         for (int i = 0; i < arInputPoints.size(); i++) {
126             for (int k = 0; k < sweepResolutionMod; k++) {
127                 xPoint3D newpt;
128                 newpt.x = arInputPoints[i].x * cos(radian) + arInputPoints[i].z * sin(radian);
129                 newpt.y = arInputPoints[i].y;
130                 newpt.z = -arInputPoints[i].x * sin(radian) + arInputPoints[i].z * cos(radian);
131
132                 arRotPoints.push_back(newpt);
133
134                 cout << "(" << newpt.x << ", " << newpt.y << ", " << newpt.z << ")\n";
135
136                 glColor3f(1.0, 1.0, 0.0);
137                 glPointSize(5.0);
138                 glBegin(GL_POINTS);
139                 glVertex3f(newpt.x, newpt.y, newpt.z);
140                 glEnd();
141                 glFlush();
142
143                 fNewAngle += fRotAngle; // 매 반복마다 새로운 각도 업데이트
144                 radian = fNewAngle * (M_PI / 180.); // 매 반복마다 라디안 업데이트
145             }
146         }

```

```

147
148 pnum = arRotPoints.size(); // point 개수 지정
149 fnum = sweepResolutionMod * (arInputPoints.size() - 1) * 2; // face 개수 지정
150 mpoint = new Point[pnum];
151 mface = new Face[fnum];
152
153 for (int i = 0; i < pnum; i++) {
154     mpoint[i].x = arRotPoints[i].x * 0.7;
155     mpoint[i].y = arRotPoints[i].y * 0.7;
156     mpoint[i].z = arRotPoints[i].z * 0.7;
157 }
158
159 for (int k = 0; k < arInputPoints.size() - 1; k++) {
160     for (int i = 0; i < sweepResolutionMod - 1; i++) { // up-triangle
161         mface[cnum].ip[1] = sweepResolutionMod * k + i + 1;
162         mface[cnum].ip[2] = sweepResolutionMod * k + i;
163         mface[cnum].ip[0] = sweepResolutionMod * (k + 1) + i;
164         cnum += 1;
165     }
166     mface[cnum].ip[1] = sweepResolutionMod * k;
167     mface[cnum].ip[2] = sweepResolutionMod * (k + 1) - 1;
168     mface[cnum].ip[0] = sweepResolutionMod * (k + 2) - 1;
169     cnum += 1;
170
171     for (int i = 0; i < sweepResolutionMod - 1; i++) { // down-triangle
172         mface[cnum].ip[0] = sweepResolutionMod * k + i + 1;
173         mface[cnum].ip[1] = sweepResolutionMod * (k + 1) + i;
174         mface[cnum].ip[2] = sweepResolutionMod * (k + 1) + i + 1;
175         cnum += 1;
176     }
177     mface[cnum].ip[0] = sweepResolutionMod * (k);
178     mface[cnum].ip[1] = sweepResolutionMod * (k + 2) - 1;
179     mface[cnum].ip[2] = sweepResolutionMod * (k + 1);
180     cnum += 1;
181 }
182 SaveModel();
183 }
184
185 if (clean) { // 화면 초기화
186     arInputPoints.clear();
187     arRotPoints.clear();
188     glClearColor(0.0, 0.0, 0.0, 0.0, 1.0);
189     glClear(GL_COLOR_BUFFER_BIT);
190     glColor3f(0.5, 0.5, 0.5); // 축 그리기
191     glBegin(GL_LINES);
192     glVertex3f(500.0, 0.0, 0.0); // x축
193     glVertex3f(-500.0, 0.0, 0.0);
194     glVertex3f(0.0, -500.0, 0.0); // y축
195     glVertex3f(0.0, 500.0, 0.0);
196     glVertex3f(0.0, 0.0, -500.0); // z축
197     glVertex3f(0.0, 0.0, 500.0);
198     glEnd();
199     glFlush();
200 }
201
202 if (Wireframe) { // wireframe 모드
203     for (int i = 0; i < arInputPoints.size(); i++) { // arInputPoints 갯수만큼 행 그리기
204         line(i * sweepResolutionMod);
205     }
206
207     for (int i = 0; i < (arInputPoints.size() - 1); i++) { // arInputPoints 갯수 - 1만큼 행 그리기
208         cross(i);
209     }
210 }
211 }
212
213
214 void Drawing(GLint button, GLint action, GLint xMouse, GLint yMouse) {
215
216     if (button == GLUT_LEFT_BUTTON && action == GLUT_DOWN && draw == true) { // 마우스 위치 입력 및 점 찍기
217         pt.x = xMouse - winWidth / 2;
218         pt.y = -yMouse + winHeight / 2;
219         pt.z = 0.0;

```

```

220         arInputPoints.push_back(pt);
221         cout << "(" << pt.x << ", " << pt.y << ", " << pt.z << ")\n";
222
223         glColor3f(1.0, 1.0, 0.0);
224         glPointSize(5.0);
225         glBegin(GL_POINTS);
226         glVertex3f(pt.x, pt.y, pt.z);
227         glEnd();
228         glFlush();
229     }
230     glutPostRedisplay();
231     glutSwapBuffers();
232 }
233
234
235 void MyMainMenu(int entryID) {
236     if (entryID == 1) { // 모든 점 지우기
237         revolveFcn = false;
238         clean = true;
239         Wireframe = false;
240         MyDisplay();
241         clean = false;
242         draw = true;
243     }
244
245     else if (entryID == 2) { // wireframe 모드
246         Wireframe = true;
247         draw = false;
248         revolveFcn = false;
249     }
250     glutPostRedisplay();
251 }
252
253
254 void MySubMenu(int entryID) {
255     if (entryID == 1) { // 30도 회전
256         cout << "(" << pt.x << ", " << pt.y << ", 0.0) by 30 radians: \n ";
257         fRotAngle = 30;
258         radian = fRotAngle * (M_PI / 180.);
259         sweepResolutionMod = 360 / fRotAngle;
260     }
261
262     else if (entryID == 2) { // 60도 회전
263         cout << "(" << pt.x << ", " << pt.y << ", 0.0) by 60 radians: \n ";
264         fRotAngle = 60;
265         radian = fRotAngle * (M_PI / 180.);
266         sweepResolutionMod = 360 / fRotAngle;
267     }
268
269     else if (entryID == 3) { // 90도 회전
270         cout << "(" << pt.x << ", " << pt.y << ", 0.0) by 90 radians: \n ";
271         fRotAngle = 90;
272         radian = fRotAngle * (M_PI / 180.);
273         sweepResolutionMod = 360 / fRotAngle;
274     }
275
276     else if (entryID == 4) { // 120도 회전
277         cout << "(" << pt.x << ", " << pt.y << ", 0.0) by 120 radians: \n ";
278         fRotAngle = 120;
279         radian = fRotAngle * (M_PI / 180.);
280         sweepResolutionMod = 360 / fRotAngle;
281     }
282
283     else if (entryID == 5) { // 180도 회전
284         cout << "(" << pt.x << ", " << pt.y << ", 0.0) by 180 radians: \n ";
285         fRotAngle = 180;
286         radian = fRotAngle * (M_PI / 180.);
287         sweepResolutionMod = 360 / fRotAngle;
288     }
289     draw = false;
290     revolveFcn = true;
291     glutPostRedisplay();
292 }
293
294 int main(int argc, char** argv) {
295     glutInit(&argc, argv);
296     glutInitDisplayMode(GLUT_RGB);
297     glutInitWindowSize(winWidth, winHeight);
298     glutInitWindowPosition(0, 0);
299     glutCreateWindow("YANGSOYEONG");
300
301     glClearColor(0.0, 0.0, 0.0, 1.0);
302     glClear(GL_COLOR_BUFFER_BIT);
303     glMatrixMode(GL_PROJECTION);
304     glLoadIdentity();
305     glOrtho(-500.0, 500.0, -350.0, 350.0, -500.0, 500.0);
306
307     GLint MySubMenuID = glutCreateMenu(MySubMenu);
308     glutAddMenuEntry("30", 1);
309     glutAddMenuEntry("60", 2);
310     glutAddMenuEntry("90", 3);
311     glutAddMenuEntry("120", 4);
312     glutAddMenuEntry("180", 5);
313
314     GLint MyMainMenuID = glutCreateMenu(MyMainMenu);
315     glutAddSubMenu("Revolve angle", MySubMenuID);
316     glutAddMenuEntry("Remove All", 1);
317     glutAddMenuEntry("Wireframe Mode", 2);
318     glutAttachMenu(GLUT_RIGHT_BUTTON);
319     glutDisplayFunc(MyDisplay);
320     glutMouseFunc(Drawing);
321     glutMainLoop();
322     return 0;
323 }

```

다음은 가상 공간 렌더링을 구현한 총 소스코드이다.

```
1  #define _CRT_SECURE_NO_WARNINGS
2
3  #include<GL/glut.h>
4  #include<stdio.h>
5  #include<math.h>
6  #include<iostream>
7
8  using namespace std;
9
10 #define WIRE 0
11 #define SHADE 1
12
13 typedef struct {
14     float x;
15     float y;
16     float z;
17 } Point;
18
19 typedef struct {
20     unsigned int ip[3];
21 } Face;
22
23 int pnum;
24 int fnum;
25 Point* mpoint = NULL;
26 Face* mface = NULL;
27 Point norm;
28
29 GLfloat angle = 0;
30
31 int moving;
32 int mousebegin;
33 int light_moving;
34 float scalefactor = 1.0;
35 int scaling = 0;
36 int status = 0;
37
38 float xmove = 500.0;
39 float ymove = 500.0;
40 float zmove = 500.0;
41 float down = 0.0;
42
43 float colorR = 0.3;
44 float colorG = 0.3;
45 float colorB = 0.3;
46
47 string fname = "c:\\data\\myModel.dat";
48
49 void Initlight() {
50     GLfloat mat_diffuse[] = { 0.5, 0.4, 0.3, 1.0 };
51     GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
52     GLfloat mat_ambient[] = { 0.5, 0.4, 0.3, 1.0 };
53     GLfloat mat_shininess[] = { 50.0 };
54     GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
55     GLfloat light_diffuse[] = { 0.8, 0.8, 0.8, 1.0 };
56     GLfloat light_ambient[] = { 0.3, 0.3, 0.3, 1.0 };
57     GLfloat light_position[] = { 200, 200, -200.0, 0.0 };
58
59     glShadeModel(GL_SMOOTH);
60     glEnable(GL_LIGHTING);
61     glEnable(GL_LIGHT0);
62     glLightfv(GL_LIGHT0, GL_POSITION, light_position);
63     glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
64     glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
65     glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
66     glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
67     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
68     glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
69     glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
70 }
71
72 Point cnormal(Point a, Point b, Point c) { // flat shading
73     Point p, q, r;
74     double val;
```

```

75     p.x = a.x - b.x; p.y = a.y - b.y; p.z = a.z - b.z;
76     q.x = c.x - b.x; q.y = c.y - b.y; q.z = c.z - b.z;
77
78     r.x = p.y * q.z - p.z * q.y;
79     r.y = p.z * q.x - p.x * q.z;
80     r.z = p.x * q.y - p.y * q.x;
81
82     val = sqrt(r.x * r.x + r.y * r.y + r.z * r.z);
83     r.x = r.x / val; r.y = r.y / val; r.z = r.z / val;
84     return r;
85 }
86
87 void ReadModel() {
88     FILE* f1;
89     char s[81];
90     int i;
91
92     if (mpoint != NULL)
93         delete mpoint;
94     if (mface != NULL)
95         delete mface;
96
97     if ((f1 = fopen(fname.c_str(), "rt")) == NULL) { printf("No file\n"); exit(0); }
98     fscanf(f1, "%s", s);
99     printf("%s", s);
100    fscanf(f1, "%s", s);
101    printf("%s", s);
102    fscanf(f1, "%d", &pnum);
103    printf("%d\n", pnum);
104    mpoint = new Point[pnum];
105
106    for (i = 0; i < pnum; i++) {
107        fscanf(f1, "%f", &mpoint[i].x);
108        fscanf(f1, "%f", &mpoint[i].y);
109        fscanf(f1, "%f", &mpoint[i].z);
110        printf("%f %f %f\n", mpoint[i].x, mpoint[i].y, mpoint[i].z);
111    }
112
113    fscanf(f1, "%s", s);
114    printf("%s", s);
115    fscanf(f1, "%s", s);
116    printf("%s", s);
117    fscanf(f1, "%d", &fnm);
118    printf("%d\n", fnm);
119
120    mface = new Face[fnm];
121    for (i = 0; i < fnm; i++) {
122        fscanf(f1, "%d", &mface[i].ip[0]);
123        fscanf(f1, "%d", &mface[i].ip[1]);
124        fscanf(f1, "%d", &mface[i].ip[2]);
125        printf("%d %d %d\n", mface[i].ip[0], mface[i].ip[1], mface[i].ip[2]);
126    }
127    fclose(f1);
128 }
129
130 void axis() {
131     glLineWidth(2);
132     glBegin(GL_LINES);
133     glColor3f(1.0, 0.0, 0.0); // draw x-axis
134     glVertex3f(0.0, -100.0, 0.0);
135     glVertex3f(160.0, -100.0, 0.0);
136     glColor3f(0.0, 1.0, 0.0); // draw y-axis
137     glVertex3f(0.0, -100.0, 0.0);
138     glVertex3f(0.0, 60.0, 0.0);
139     glColor3f(0.0, 0.0, 1.0); // draw z-axis
140     glVertex3f(0.0, -100.0, 0.0);
141     glVertex3f(0.0, -100.0, 160.0);
142     glEnd();
143     glLineWidth(1.0);
144 }
145
146 void DrawWire(void) {
147     glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
148     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

```



```

149     glEnable(GL_DEPTH_TEST);
150     axis();
151     glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
152     glCallList(1);
153
154     glColor3f(0.7, 0.7, 0.7);
155     glPushMatrix();
156     glTranslatef(0, -100, 0);
157     glBegin(GL_QUADS);
158     glVertex3f(200, 0, 200);
159     glVertex3f(200, 0, -200);
160     glVertex3f(-200, 0, -200);
161     glVertex3f(-200, 0, 200);
162     glEnd();
163     glPopMatrix();
164     glutSwapBuffers();
165 }
166
167 void DrawShade(void) {
168     glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
169     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
170     glEnable(GL_DEPTH_TEST);
171     axis();
172     glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
173     glCallList(1);
174
175     glColor3f(0.6, 0.6, 0.6); // draw plane
176     glPushMatrix();
177     glTranslatef(0, -100, 0);
178     glBegin(GL_QUADS);
179     glVertex3f(200, 0, 200);
180     glVertex3f(200, 0, -200);
181     glVertex3f(-200, 0, -200);
182     glVertex3f(-200, 0, 200);
183     glEnd();
184     glPopMatrix();
185     glutSwapBuffers();
186 }
187
188 void MakeGL_Model(void) {
189     int i;
190     glShadeModel(GL_SMOOTH);
191
192     if (glIsList(1)) glDeleteLists(1, 1);
193     glNewList(1, GL_COMPILE);
194     glPushMatrix();
195
196     glRotatef(angle, 0.0, 1.0, 0.0);
197     glScalef(scalefactor, scalefactor, scalefactor);
198     glColor3f(1, 0, 0);
199
200     glColor3f(colorR, colorG, colorB);
201
202     glEnable(GL_NORMALIZE);
203     for (i = 0; i < fnum; i++) {
204         Point norm = mpoint[mface[i].ip[0]];
205         glBegin(GL_TRIANGLES);
206
207         glNormal3f(norm.x, norm.y, norm.z); // 면을 이루는 첫번째 vertex
208         glVertex3f(mpoint[mface[i].ip[0]].x, mpoint[mface[i].ip[0]].y, mpoint[mface[i].ip[0]].z);
209
210         norm = mpoint[mface[i].ip[1]];
211         glNormal3f(norm.x, norm.y, norm.z); // 면을 이루는 두번째 vertex
212         glVertex3f(mpoint[mface[i].ip[1]].x, mpoint[mface[i].ip[1]].y, mpoint[mface[i].ip[1]].z);
213
214         norm = mpoint[mface[i].ip[2]];
215         glNormal3f(norm.x, norm.y, norm.z); // 면을 이루는 세번째 vertex
216         glVertex3f(mpoint[mface[i].ip[2]].x, mpoint[mface[i].ip[2]].y, mpoint[mface[i].ip[2]].z);
217         glEnd();
218     }
219     glPopMatrix();
220     glEndList();
221 }
222

```

```

223 void perspmode() { // perspective mode
224     glEnable(GL_DEPTH_TEST);
225     glMatrixMode(GL_PROJECTION);
226     glLoadIdentity();
227     gluPerspective(40.0, 1.0, 1.0, 2000.0);
228     glMatrixMode(GL_MODELVIEW);
229     glLoadIdentity();
230     gluLookAt(600.0, 600.0, 600.0, 0.0, 0.0, 0.0, 1.0, 0.);
231 }
232
233 void orthomode() { // ortho mode
234     glEnable(GL_DEPTH_TEST);
235     glMatrixMode(GL_PROJECTION);
236     glLoadIdentity();
237     glOrtho(-100.0, 100.0, -100.0, 100.0, -800.0, 800.0);
238     glMatrixMode(GL_MODELVIEW);
239     glLoadIdentity();
240     gluLookAt(600.0, 600.0, 600.0, 0.0, 0.0, 0.0, 1.0, 0.);
241 }
242
243 void GLSetupRC(void) {
244     glEnable(GL_DEPTH_TEST);
245     glMatrixMode(GL_PROJECTION);
246     glLoadIdentity();
247     gluPerspective(40.0, 1.0, 1.0, 2000.0);
248     glMatrixMode(GL_MODELVIEW);
249     glLoadIdentity();
250     glutPostRedisplay();
251 }
252
253 void display(void) {
254     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
255     glLoadIdentity();
256     gluLookAt(xmove, ymove, zmove, 0.0, 0.0, 0.0, 1.0, down);
257     MakeGL_Model();
258     if (status == WIRE)
259         DrawWire();
260     else
261         DrawShade();
262 }
263
264 void keyboard(unsigned char key, int x, int y) {
265     printf("key %d\n", key);
266     switch (key) {
267     case 'w': // wire mode
268         status = WIRE;
269         glutPostRedisplay();
270         break;
271     case 's': // shade mode
272         status = SHADE;
273         glutPostRedisplay();
274         break;
275
276     case 'e': // hide back
277         glEnable(GL_CULL_FACE);
278         glCullFace(GL_BACK);
279         glutPostRedisplay();
280         break;
281     case 'd': // show back
282         glDisable(GL_CULL_FACE);
283         glutPostRedisplay();
284         break;
285
286     case 'r': // perspective mode
287         display();
288         perspmode();
289         glutPostRedisplay();
290         break;
291     case 'f': // ortho mode
292         display();
293         orthomode();
294         glutPostRedisplay();
295         break;
296

```

```

297     case 'q': // x viewport
298         xmove = 600;
299         ymove = 0;
300         zmove = 0;
301         down = 0;
302         display();
303         glMatrixMode(GL_MODELVIEW);
304         glLoadIdentity();
305         glutPostRedisplay();
306         break;
307     case 'a': // y viewports
308         xmove = 0;
309         ymove = 600;
310         zmove = 0;
311         down = -1;
312         display();
313         glMatrixMode(GL_MODELVIEW);
314         glLoadIdentity();
315         glutPostRedisplay();
316         break;
317     case 'z': // z viewport
318         xmove = 0;
319         ymove = 0;
320         zmove = 600;
321         down = 0;
322         display();
323         glMatrixMode(GL_MODELVIEW);
324         glLoadIdentity();
325         glutPostRedisplay();
326         break;
327
328     case 'l': // red color
329         colorR = 0.8; colorG = 0.4; colorB = 0.4;
330         display();
331         glutPostRedisplay();
332         break;
333     case '2': // green color
334         colorR = 0.4; colorG = 0.8; colorB = 0.4;
335         display();
336         glutPostRedisplay();
337         break;
338     case '3': // blue color
339         colorR = 0.4; colorG = 0.4; colorB = 0.8;
340         display();
341         glutPostRedisplay();
342         break;
343
344     case 'b': // xmoving
345         xmove += 10;
346         display();
347         glutPostRedisplay();
348         break;
349     case 'n': // ymoving
350         ymove += 10;
351         display();
352         glutPostRedisplay();
353         break;
354     case 'm': // zmoving
355         zmove += 10;
356         display();
357         glutPostRedisplay();
358         break;
359     case 'h': // xmoving
360         xmove -= 10;
361         display();
362         glutPostRedisplay();
363         break;
364     case 'j': // ymoving
365         ymove -= 10;
366         display();
367         glutPostRedisplay();
368         break;
369     case 'k': // zmoving
370         zmove -= 10;

```

```

371         display();
372         glutPostRedisplay();
373         break;
374     }
375 }
376
377 void mouse(int button, int state, int x, int y) {
378     if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
379         moving = 1;
380         mousebegin = x;
381     }
382     if (button == GLUT_LEFT_BUTTON && state == GLUT_UP) {
383         moving = 0;
384     }
385     if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) {
386         scaling = 1;
387         mousebegin = x;
388     }
389     if (button == GLUT_RIGHT_BUTTON && state == GLUT_UP) {
390         scaling = 0;
391     }
392 }
393
394 void motion(int x, int y) {
395     if (scaling) {
396         scalefactor = scalefactor * (1.0 + (mousebegin - x) * 0.0001);
397         glutPostRedisplay();
398     }
399     if (moving) {
400         angle = angle + (x - mousebegin);
401         mousebegin = x;
402         glutPostRedisplay();
403     }
404 }
405
406 int main(int argc, char** argv) {
407     glutInit(&argc, argv);
408     glutInitWindowSize(500, 500); glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
409     glutInitWindowPosition(100, 100);
410     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
411     glutCreateWindow("YANGSOYEONG");
412     glutDisplayFunc(display);
413     glutKeyboardFunc(keyboard);
414     glutMouseFunc(mouse);
415     glutMotionFunc(motion);
416     InitLight();
417     ReadModel();
418     GLSetupRC();
419     glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);
420     glEnable(GL_COLOR_MATERIAL);
421     glutMainLoop();
422     return 0;
423 }

```

3. 실행 결과

