(중간고사 대체 프로젝트)

컴퓨터 그래픽스

- OpenGL을 이용한 SOR Modeling -

중앙대학교 예술공학대학 컴퓨터예술학부

20194004

양소영

1. 문제 내용 및 해결 방안

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

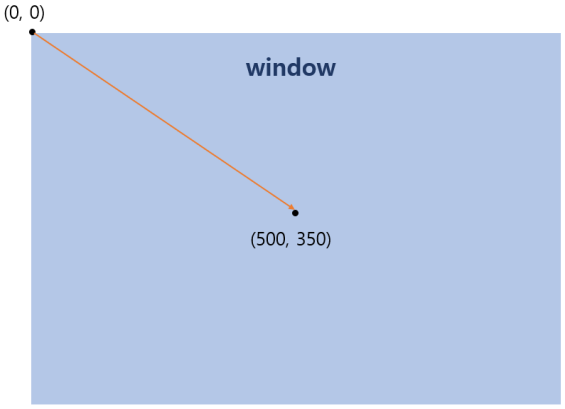
Window 생성> 화면에 점 찍기> 메뉴 생성> 점 회전> 화면 초기화> wireframe 모드 생성

(1) Window 생성

먼저 window 생성 단계이다. Window 크기는 (1000, 700)으로 지정하였고 x 축 및 y 축 좌표축은 GL_LINES 로 그려주었다.

(2) 화면에 점 찍기

화면에 점을 찍기 위해서는 [그림 1]과 같이 window 의 원점을 화면 정중앙으로 옮겨주는 과정이 필요한데, 이를 구현하는 방법은 마우스로 찍은 점의 위치를 변경하여 출력하는 것이다. 예를 들어 (0, 0)에 점을 찍으면 일정한 값을 곱하거나 더해주어 (500, 350)의 위치로 반환시킨다. 본인의 window 의 크기는 1000x700 이므로 마우스의 위치 (x, y)에서 (x–winWidth/2, -y +winHeight/2)으로 변경해주었다.



[그림 1]

점을 찍는 방법은 교안을 참고하였다. 마우스 왼쪽 버튼(GLUT_LEFT_BUTTON)을 누를 때 (GLUT_DOWN) glBegin(GL_POINTS)를 사용하여 점을 찍을 수 있도록 하였다. 처음에는 2 차원의 공간에서 구현하였지만, 점을 회전하기 위해서는 3 차원의 공간이 필요하기 때문에 glOrtho 함수를 입력하여 직교 투영이 가능하도록 만들어준다. 윈도우 전체를 투영해야 되므로 glOrtho 의 파라미터 순서는 (-500.0, 500.0, -350.0, 350.0, -500.0, 500.0)으로 지정해주었다. 마지막 두 파라미터 near 과 far 의 파라미터에 충분히 큰 값을 입력하지 않으면 회전한 점이 화면 상에 보이지 않으므로 주의한다.

(3) 메뉴 생성

다음은 메뉴를 생성하는 과정이다. 이 방법 또한 교안을 참고하였다. Window 화면에서 마우스 오른쪽 버튼(GLUT_RIGHT_BUTTON)을 클릭하면 메뉴가 뜨도록 하였다. 메인 메뉴 순으로 'Revolve angle (회전하는 각도)', 'Remove All (화면 초기화)', 'Wireframe Mode (와이어프레임 모드)'를 생성하였다. 또한 첫번째 메인 메뉴에서는 30 도, 60 도, 90 도, 120 도, 180 도와 같이 각도를 여러가지 옵션 중에서 선택할 수 있도록 서브 메뉴를 추가하였다.

(4) 점 회전

Window 화면에서 점을 여러 개 찍고 마우스 오른쪽 버튼을 클릭하여 메인 메뉴가 나타나면, 원하는 각도를 눌렀을 때 회전한 점들이 화면에 나타나도록 구현하고자 하였다. 이를 위해 GLboolean 타입의 변수가 필요하였다. 디폴트 값이 false 인 'revolveFcn' 이라는 boolean 형 타입의 변수를 만들어서 원하는 각도를 누르면 boolean 을 true 로 변경해 입력 받은 점을 회전시켜 vector 에 저장하고, 저장한 모든 점들을 나타내도록 하였다.

각도에 따라 변수에 넣는 값이 달라지기 때문에 입력 받을 각도 fRotAngle, 라디안 radian, 점이 회전할 횟수 sweepResolutionMod 는 모두 전역변수로 선언하였다. 또한 점의 위치를 나타낼 x, y, z 좌표는 계속 쓰일 수 있도록 class 로 정의해주었다. 그리고 그 클래스를 vector 로 받아 초기에 찍은 점을 받는 arInputPoints, 회전한 점을 추가해주도록 하는 arRotPoints 를 선언하였다.

처음에 window 에서 마우스로 입력하는 점들은 arInputPoints 에 저장해 놓고, 저장한 그 점들을 sweepResolutionMod 값만큼 회전시켜 다시 arRotPoints 에 저장해주도록 하였다. 이 때 이중 for 문이 필요한데, 첫번째 반복문은 입력한 점 개수만큼, 두번째 반복문은 점이 회전해야 하는 횟수만큼 반복하도록 입력하였다.

점을 y 축으로 회전하기 위해서는 y 축 변환 행렬이 필요하다. 교안에 있는 변환 행렬이 잘못 나와있어서 점이 계속 이상하게 출력되어 오랜 시간동안 고민을 하였다. 다음은 제대로 구한 y 축 변환 행렬이다.

$$x' = x\cos\theta + z\sin\theta$$
$$y' = y$$
$$z' = -x\sin\theta + z\cos\theta$$

변환 행렬의 θ 값에는 radian 이 들어가므로 반복문의 횟수가 증가할 때마다 각도와 라디안 값이 바뀌도록 업데이트해준다. 이를 해결하기 위해 새로운 fNewAngle 라는 변수를 생성하여

0 도부터 시작해 원하는 각도만큼 증가하도록 하고, radian 에 fNewAngle 값을 넣어주도록 한다. 따라서 출력된 점들이 모두 arRotPoints 의 vector 배열에 들어가게 된다.

(5) 화면 초기화

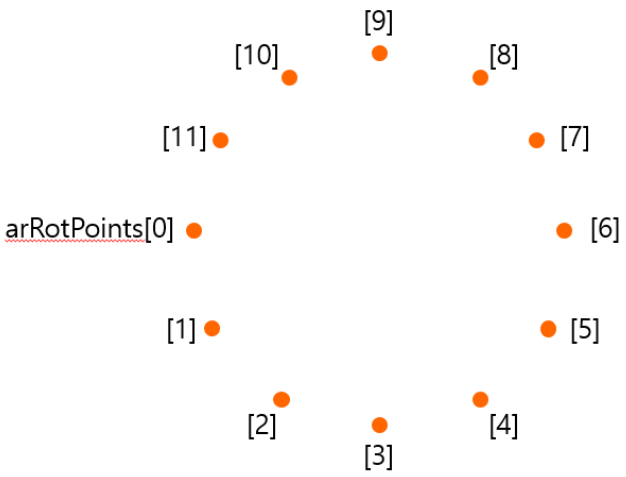
모든 점이 출력된 후 window 를 다시 초기 상태로 만들어 주기 위해 화면 초기화 메뉴가 필요하다고 생각하였다. 화면을 초기화하기 위해 clean 이라는 boolean 타입의 함수를 따로 만들어 주었는데, arInputPoints 와 arRotPoints 의 배열을 모두 clear 해주고 다시 초기 윈도우 화면으로 세팅 되도록 해주었다.

메인 메뉴에서 스크립트를 작성할 때는 clean 함수 외에 다른 모든 함수들을 false 처리 해주었다. 그리고 화면이 초기화되면 다시 점이 찍힐 수 있도록 해야 하는데, 이 문제를 해결하기 위해 디폴트 값이 true 인 draw 라는 boolean 타입 변수를 만들어 마우스 위치를 입력 받는 코드에 함께 삽입해주었다. 이로써 window 화면을 초기화하면 다시 점이 찍힐 수 있게 되었다. 이 draw boolean 은 추후에 wireframe 모드가 진행될 때 점을 더 이상 입력 받을 수 없도록 하는 역할을 한다.

(6) wireframe 모드 생성

알고리즘 적인 문제 해결을 가장 많이 요구했던 과정이라고 생각한다. Wireframe 을 만들어주는 함수를 따로 작성하느라 굉장히 머리를 많이 썼던 것 같다.

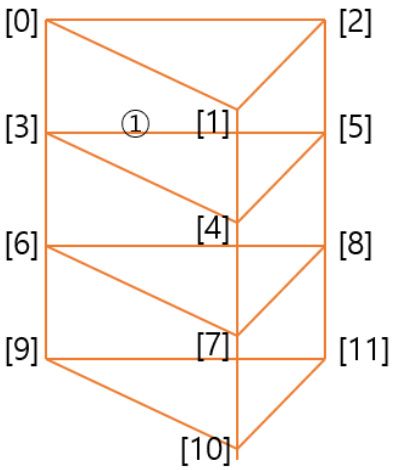
제일 처음에는 가로 방향의 wireframe 이 그려지도록 행 출력 함수 line 을 정의하였다. 예를 들어 30 도를 입력한 점은 한 줄 당 12 번이 출력되어야 한다.



[그림 2]

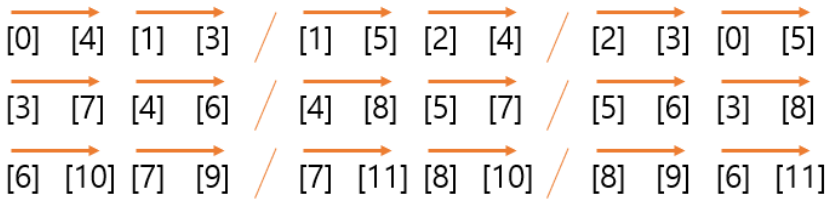
[그림 2]는 1 행의 점 배열을 위에서 내려 본 것이다. 한 행 당 sweepResolutionMod 회를 출력하기 위해 line 함수에서 for 반복문을 써주고, wireframe 의 boolean 정의에서는 행의 개수가 arInputPoints 만큼 출력되도록 for 반복문을 써준다.

가장 큰 문제는 삼각형의 wireframe 을 그려주는 함수 구현이었다. 먼저 cross 라는 함수를 정의해주었다.



[그림 3]

예를 들어 [그림 3]과 같이 처음에 4 개의 점을 입력하여 120 도만큼 돌린 입체 도형을 생성한다고 하자. 본인이 생각하는 cross 함수는 ①과 같은 각 면마다 'X'자로 선을 그려주는 것인데, 위 도형에서 cross 함수를 적용하려면 [그림 4]와 같은 방향성을 가진 선들이 필요하다.



[그림 4]

[그림 4]의 첫번째 행은 ①번과 같은 행에 있는 면들에 'X'자 선을 그어주는 것이다. 도형 전체적으로 이 행을 총 arInputPoints.size 에서 1 을 뺀 값만큼 반복해야 하기 때문에 이를 wireframe 의 boolean 정의에 작성한다. 이 반복 횟수를 cross 함수에서 c 라는 매개변수로 받도록 한다. 예시로 든 [그림 4]의 첫번째 행은 세개의 단락으로 나누어져 있다. 이 단락이 총 sweepResolutionMod 개수만큼 존재해야 하는데, 마지막 단락은 앞 두개의 단락과 다른 형식이므로 제외시켜준다.

2. 소스코드

다음은 SOR Modeling을 구현한 총 소스코드이다.

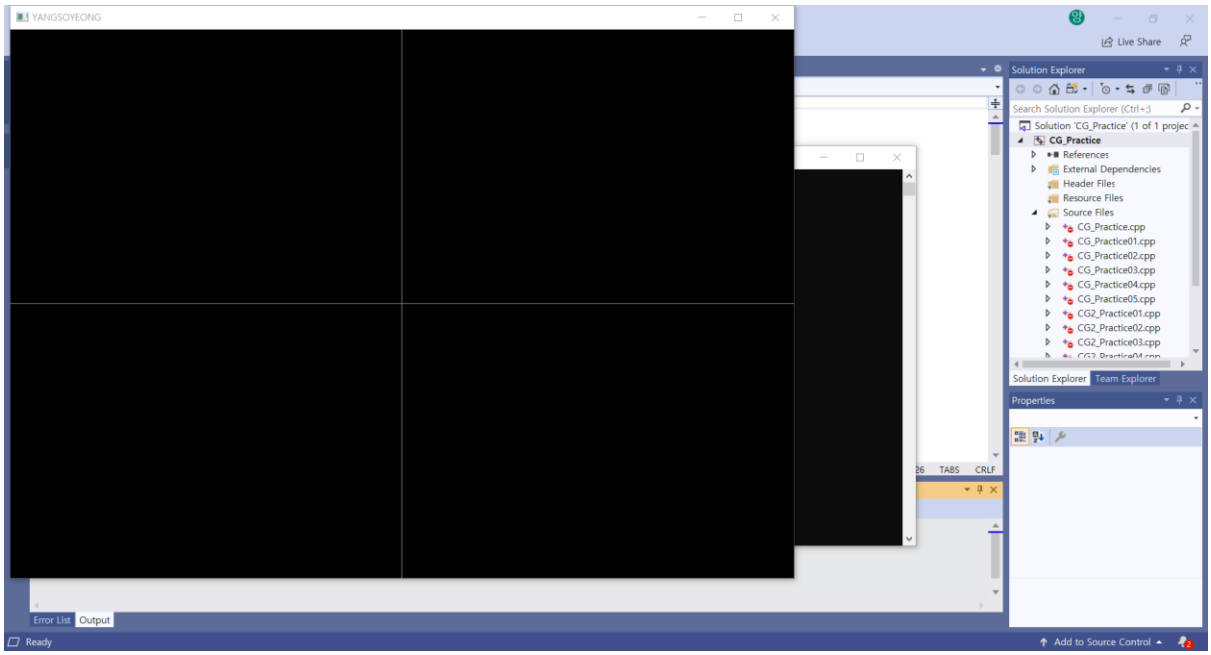
```
#define _USE_MATH_DEFINES
#include <GL/glut.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <iostream>
using namespace std;

GLsizei winWidth = 1000, winHeight = 700;
GLboolean revolveFcn = false;  // 점 회전
GLboolean clean = false;  // 화면 초기화
GLboolean draw = true;  // 점 찍기
GLboolean Wireframe = false;  // wireframe 모드

float fRotAngle;  // 입력받은 각도
float radian;
float sweepResolutionMod = (360 / fRotAngle);  // 점이 회전하는 횟수


class xPoint3D {
public:
    float x, y, z, w;
    xPoint3D() { x = y = z = 0; w = 1; };
};

xPoint3D pt;
vector<xPoint3D> arInputPoints;  // 초기에 찍은 점 받기
vector<xPoint3D> arRotPoints;  // 회전한 점 받기


void line(int l) {
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_LINE_STRIP);  // wireframe 행 그리기
    for (int i = 0; i < sweepResolutionMod; i++) {
        glVertex3f(arRotPoints[l + i].x, arRotPoints[l + i].y, arRotPoints[l + i].z);
    }
    glEnd();
    glFlush();
}


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 + 1 + i].x, arRotPoints[c * sweepResolutionMod + 1 + i].y, arRotPoints[c * sweepResolutionMod + 1 + i].z);
        glVertex3f(arRotPoints[c * sweepResolutionMod + sweepResolutionMod + i].x, arRotPoints[c * sweepResolutionMod + sweepResolutionMod + i].y,
            arRotPoints[c * sweepResolutionMod + sweepResolutionMod + 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);
    glVertex3f(arRotPoints[c * sweepResolutionMod + (2 * sweepResolutionMod - 1)].x, arRotPoints[c * sweepResolutionMod + (2 * sweepResolutionMod - 1)].y,
        arRotPoints[c * sweepResolutionMod + (2 * sweepResolutionMod - 1)].z);
    glEnd();
    glFlush();
}


void MyDisplay() {
    glViewport(0, 0, 1000, 700);
    glColor3f(1.0, 0.0, 0.0);

    glColor3f(0.5, 0.5, 0.5);  // 축 그리기
    glBegin(GL_LINES);
    glVertex3f(500.0, 0.0, 0.0);  // x축
    glVertex3f(-500.0, 0.0, 0.0);
    glVertex3f(0.0, -500.0, 0.0);  // y축
    glVertex3f(0.0, 500.0, 0.0);
    glVertex3f(0.0, 0.0, -500.0);  // z축
    glVertex3f(0.0, 0.0, 500.0);
    glEnd();

    if (revolveFcn) {  // 점 회전

        float fNewAngle = 0;
        radian = fNewAngle * (M_PI / 180.0);
```

```
        for (int i = 0; i < arInputPoints.size(); i++) {
            for (int k = 0; k < sweepResolutionMod; k++) {
                xPoint3D newpt;
                newpt.x = arInputPoints[i].x * cos(radian) + arInputPoints[i].z * sin(radian);
                newpt.y = arInputPoints[i].y;
                newpt.z = -arInputPoints[i].x * sin(radian) + arInputPoints[i].z * cos(radian);

                arRotPoints.push_back(newpt);

                cout << "(" << newpt.x << ", " << newpt.y << ", " << newpt.z << ")\n";

                glColor3f(1.0, 1.0, 0.0);
                glPointSize(5.0);
                glBegin(GL_POINTS);
                glVertex3f(newpt.x, newpt.y, newpt.z);
                glEnd();
                glFlush();

                fNewAngle += fRotAngle;  // 매 반복마다 새로운 각도 업데이트
                radian = fNewAngle * (M_PI / 180.);  // 매 반복마다 라디안 업데이트
            }
        }
    }

    if (clean) {  // 화면 초기화
        arInputPoints.clear();
        arRotPoints.clear();
        glClearColor(0.0, 0.0, 0.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0.5, 0.5, 0.5);  // 축 그리기
        glBegin(GL_LINES);
        glVertex3f(500.0, 0.0, 0.0);  // x축
        glVertex3f(-500.0, 0.0, 0.0);
        glVertex3f(0.0, -500.0, 0.0);  // y축
        glVertex3f(0.0, 500.0, 0.0);
        glVertex3f(0.0, 0.0, -500.0);  // z축
        glVertex3f(0.0, 0.0, 500.0);
        glEnd();
        glFlush();
    }

    if (Wireframe) {  // wireframe 모드
        for (int i = 0; i < arInputPoints.size(); i++) {  // arInputPoints 갯수만큼 행 그리기
            line(i * sweepResolutionMod);
        }

        for (int i = 0; i < (arInputPoints.size() - 1); i++) {
            cross(i);
        }
    }
}


void Drawing(GLint button, GLint action, GLint xMouse, GLint yMouse) {

    if (button == GLUT_LEFT_BUTTON && action == GLUT_DOWN && draw == true) {  // 마우스 위치 입력 및 점 찍기
        pt.x = xMouse - winWidth / 2;
        pt.y = -yMouse + winHeight / 2;
        pt.z = 0.0;
        arInputPoints.push_back(pt);
        cout << "(" << pt.x << ", " << pt.y << ", " << pt.z << ")\n";

        glColor3f(1.0, 1.0, 0.0);
        glPointSize(5.0);
        glBegin(GL_POINTS);
        glVertex3f(pt.x, pt.y, pt.z);
        glEnd();
        glFlush();
    }
    glutPostRedisplay();
    glutSwapBuffers();
}


void MyMainMenu(int entryID) {
    if (entryID == 1) {  // 모든 점 지우기
        revolveFcn = false;
        clean = true;
        Wireframe = false;
        MyDisplay();
        clean = false;
        draw = true;
    }

    else if (entryID == 2) {  // wireframe 모드
        Wireframe = true;
        draw = false;
        revolveFcn = false;
    }

    glutPostRedisplay();
}


void MySubMenu(int entryID) {
    if (entryID == 1) {  // 30도 회전
        cout << "(" << pt.x << ", " << pt.y << ", 0.0) by 30 radians: \n ";
        fRotAngle = 30;
        radian = fRotAngle * (M_PI / 180.);
        sweepResolutionMod = 360 / fRotAngle;
    }

    else if (entryID == 2) {  // 60도 회전
        cout << "(" << pt.x << ", " << pt.y << ", 0.0) by 60 radians: \n ";
        fRotAngle = 60;
        radian = fRotAngle * (M_PI / 180.);
```

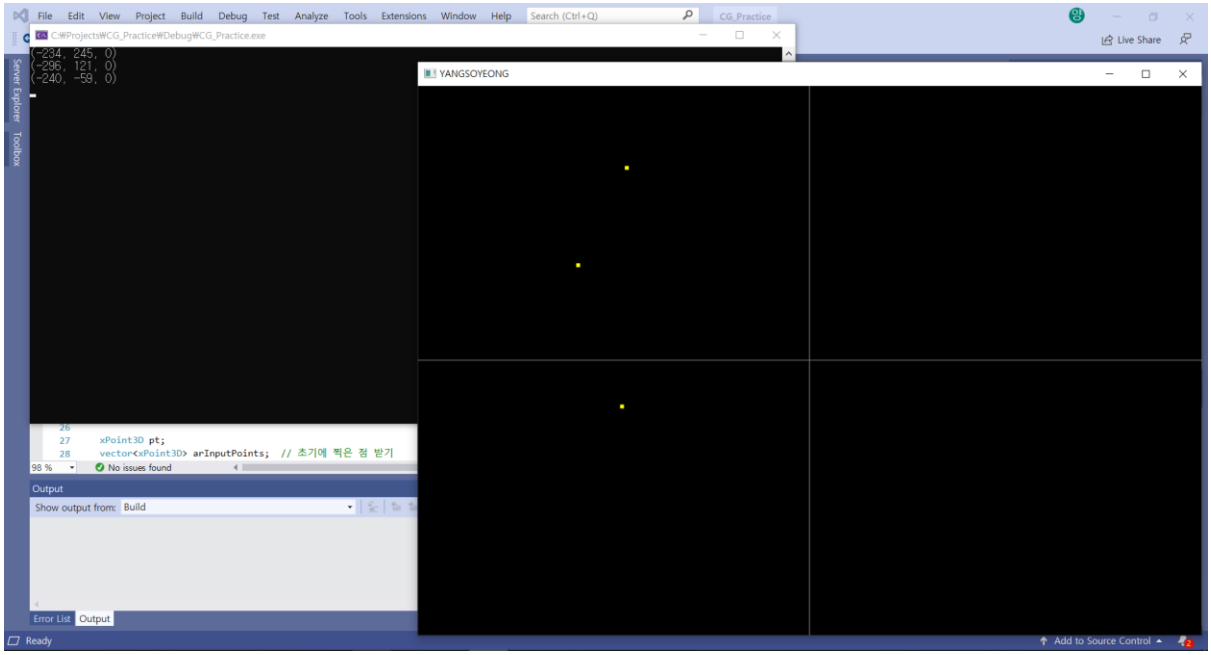
```
        sweepResolutionMod = 360 / fRotAngle;
    }

    else if (entryID == 3) {  // 90도 회전
        cout << "(" << pt.x << ", " << pt.y << ", 0.0) by 90 radians: \n ";
        fRotAngle = 90;
        radian = fRotAngle * (M_PI / 180.);
        sweepResolutionMod = 360 / fRotAngle;
    }

    else if (entryID == 4) {  // 120도 회전
        cout << "(" << pt.x << ", " << pt.y << ", 0.0) by 120 radians: \n ";
        fRotAngle = 120;
        radian = fRotAngle * (M_PI / 180.);
        sweepResolutionMod = 360 / fRotAngle;
    }

    else if (entryID == 5) {  // 180도 회전
        cout << "(" << pt.x << ", " << pt.y << ", 0.0) by 180 radians: \n ";
        fRotAngle = 180;
        radian = fRotAngle * (M_PI / 180.);
        sweepResolutionMod = 360 / fRotAngle;
    }

    draw = false;
    revolveFcn = true;
    glutPostRedisplay();

}


int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB);
    glutInitWindowSize(winWidth, winHeight);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("YANGSOYEONG");

    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-500.0, 500.0, -350.0, 350.0, -500.0, 500.0);

    GLint MySubMenuID = glutCreateMenu(MySubMenu);
    glutAddMenuEntry("30", 1);
    glutAddMenuEntry("60", 2);
    glutAddMenuEntry("90", 3);
    glutAddMenuEntry("120", 4);
    glutAddMenuEntry("180", 5);

    GLint MyMainMenuID = glutCreateMenu(MyMainMenu);
    glutAddSubMenu("Revolve angle", MySubMenuID);
    glutAddMenuEntry("Remove All", 1);
    glutAddMenuEntry("Wireframe Mode", 2);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutDisplayFunc(MyDisplay);
    glutMouseFunc(Drawing);
    glutMainLoop();
    return 0;
}
```

3. 실행 결과

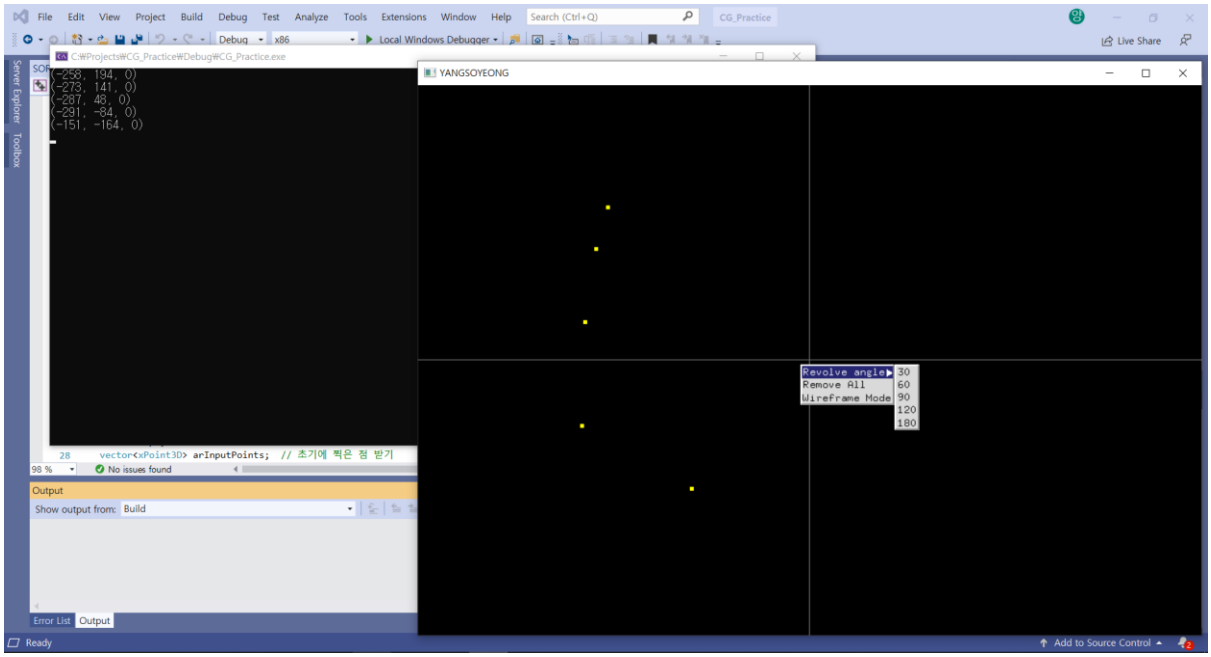
처음으로 코드를 실행시키면 윈도우 화면이 나타난다.



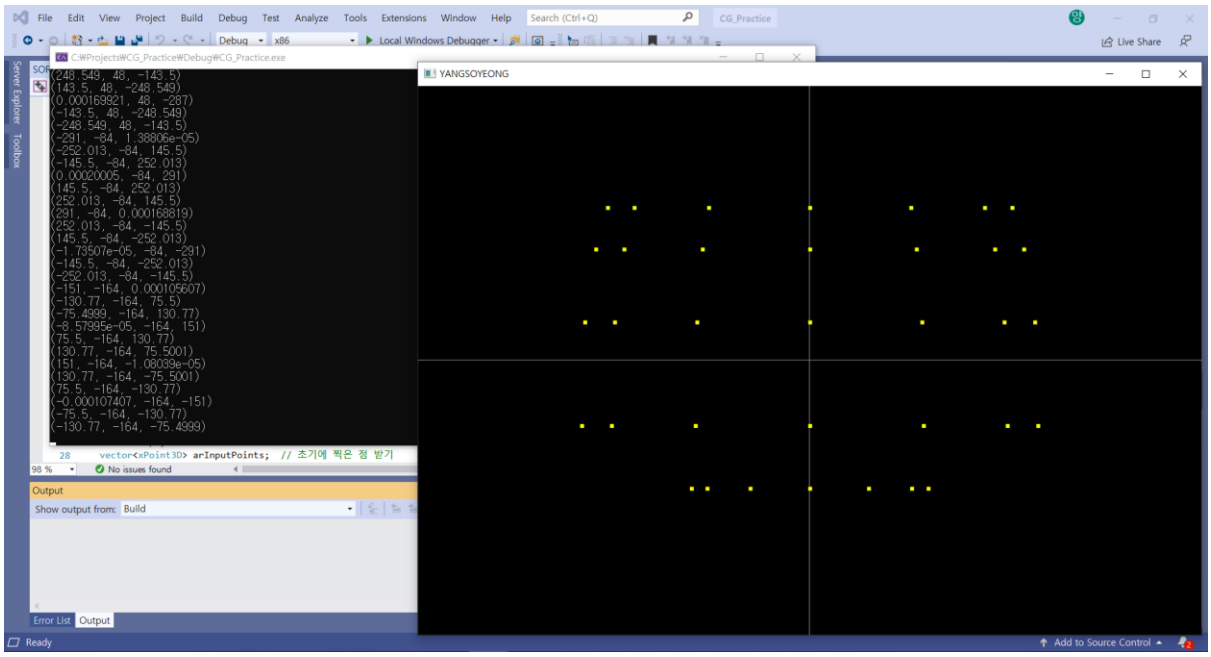
마우스 왼쪽 버튼을 누르면 점을 찍으면 해당 점의 좌표가 출력된다.



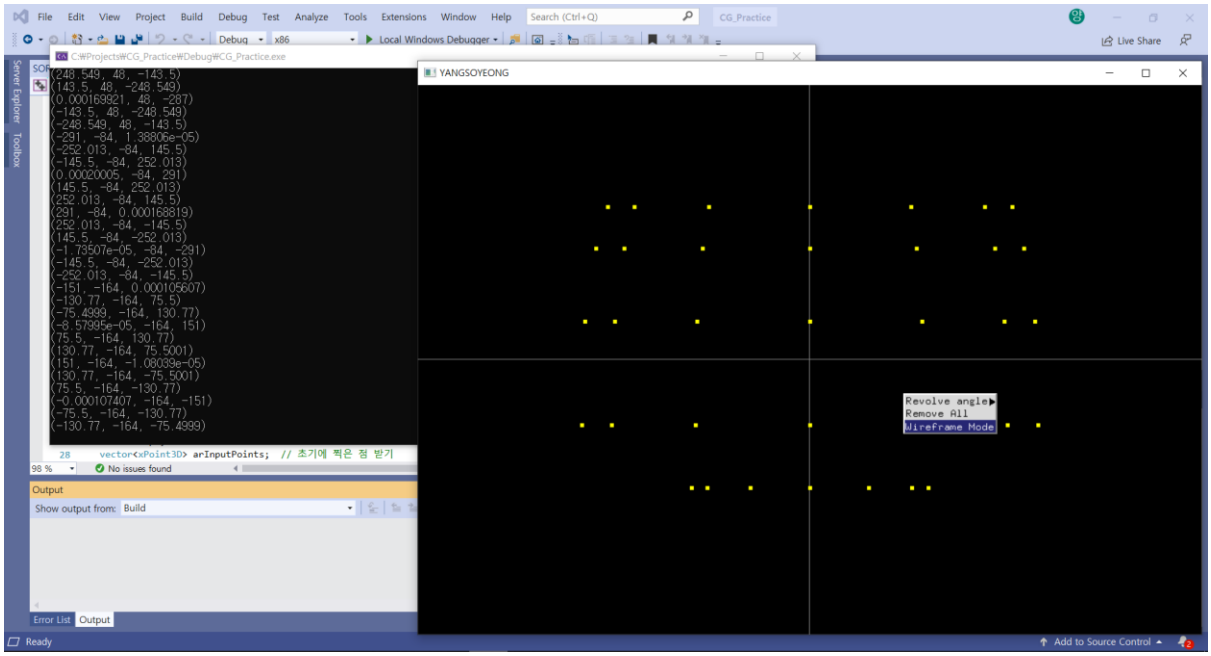
윈도우 화면에서 마우스 오른쪽 버튼을 클릭하면 메뉴가 나타난다. 첫번째 메인 메뉴에서는 회전할 각을 입력 받는다.



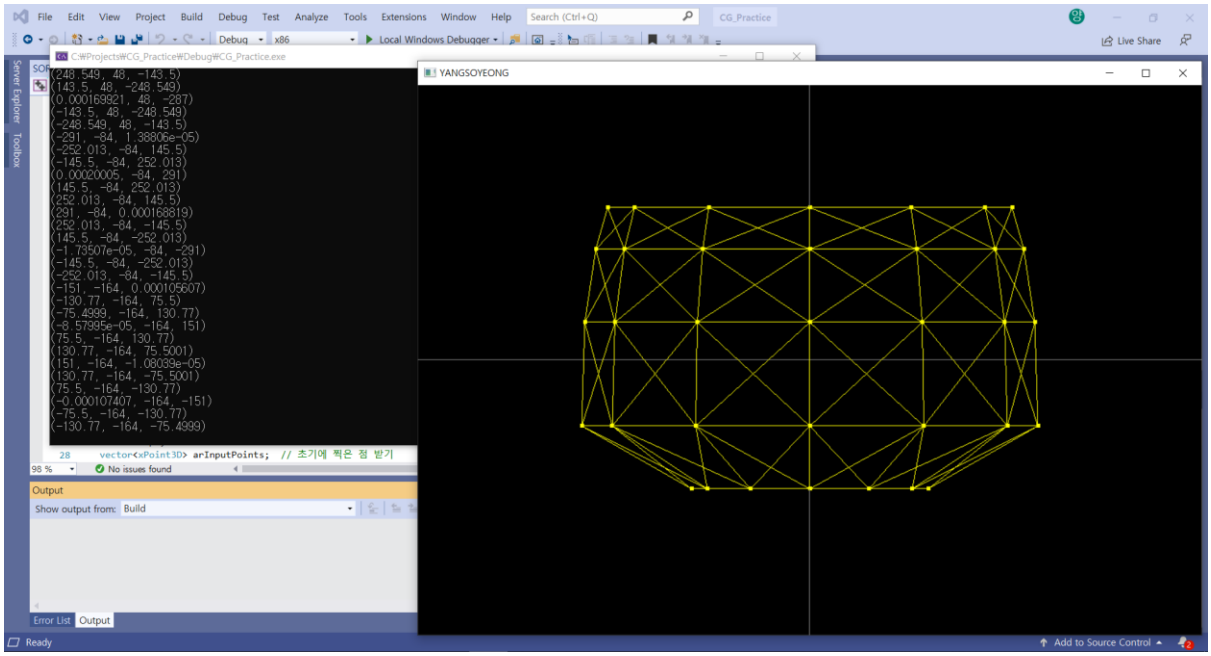
30도를 입력하면 다음과 같이 회전한 모든 점이 출력된다.



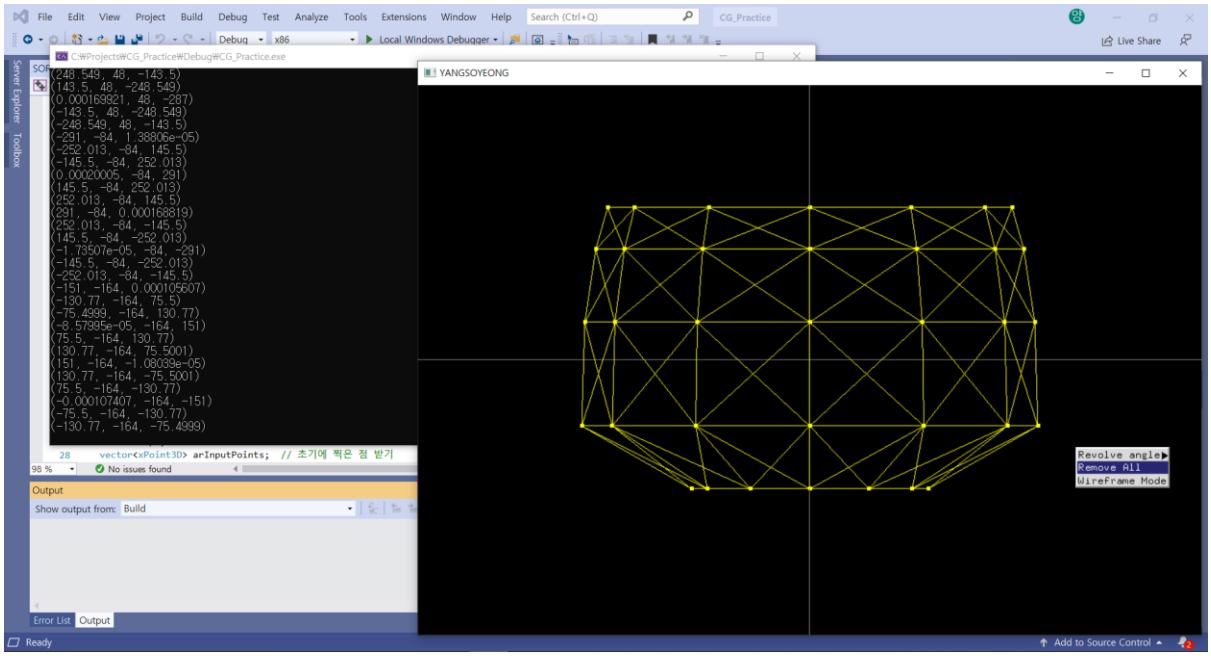
Wireframe 모드를 보고 싶으면 이 상태에서 바로 메인 메뉴의 Wireframe Mode를 선택한다.



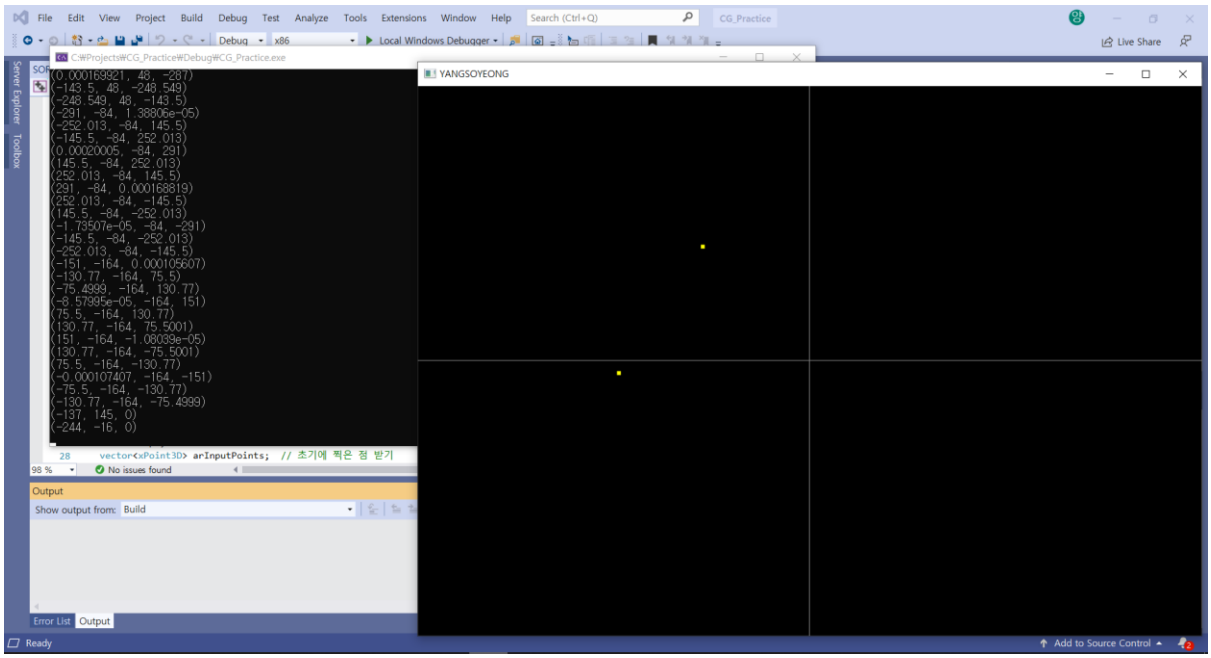
Wireframe이 나타나는 것을 볼 수 있다.



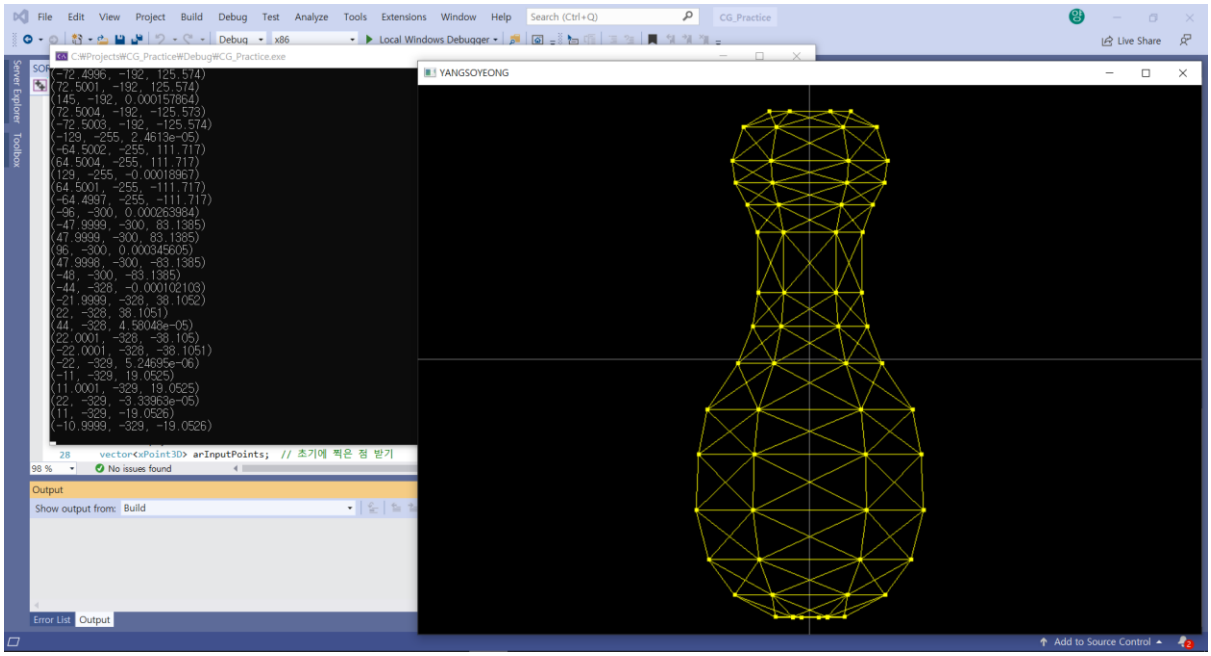
화면을 초기화하여 다시 점을 찍고 싶으면 이 상태에서 바로 메인 메뉴의 Remove All을 클릭한다.



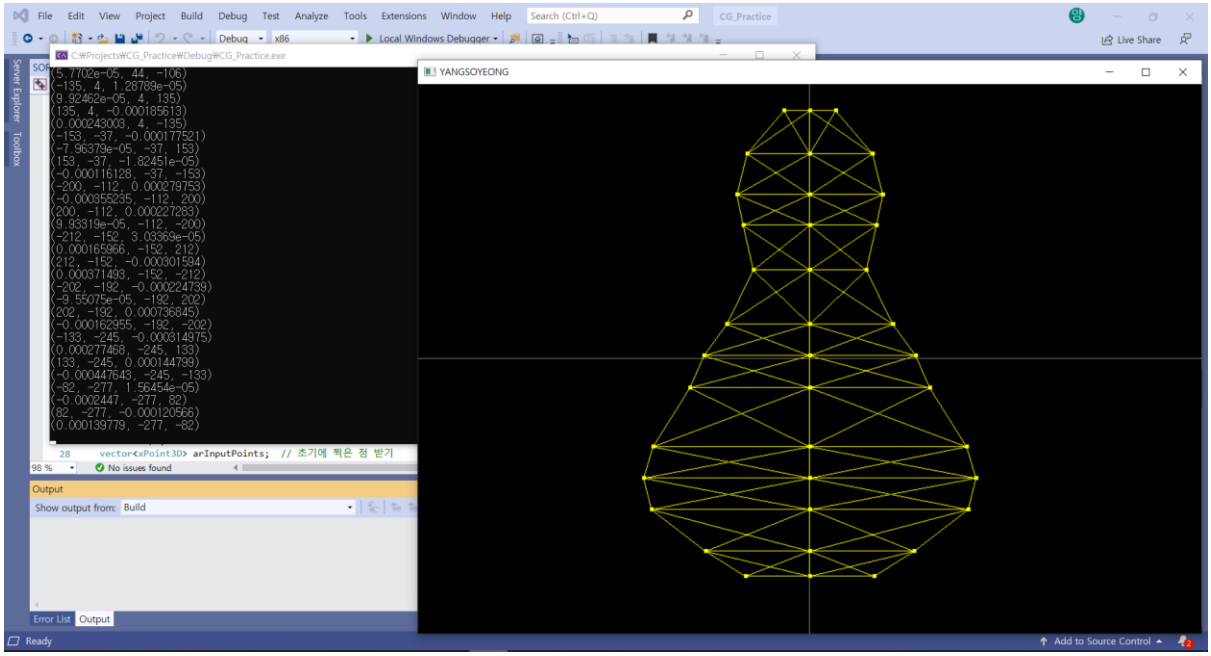
윈도우의 초기상태로 되돌아 가 화면에 다시 점을 찍는 것이 가능하게 된다.



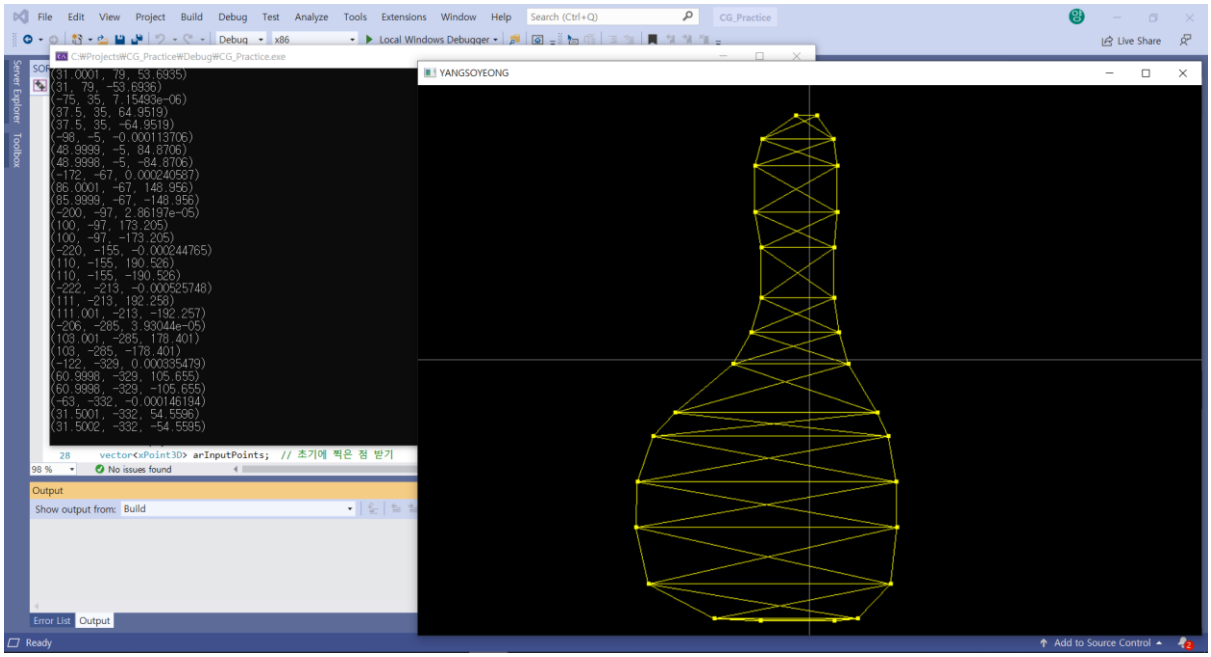
30도 외에도 60도, 90도, 120도, 180도 모두 wireframe 적용이 가능하다.



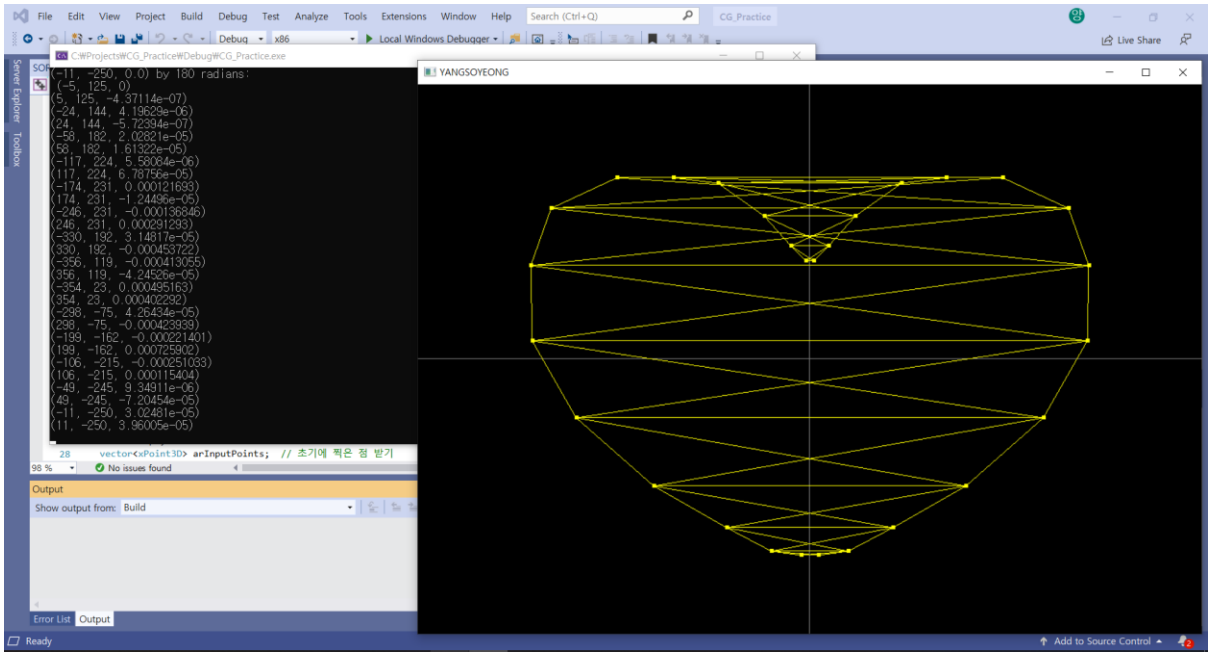
[60° 적용]



[90° 적용]



[120° 적용]



[180° 적용]