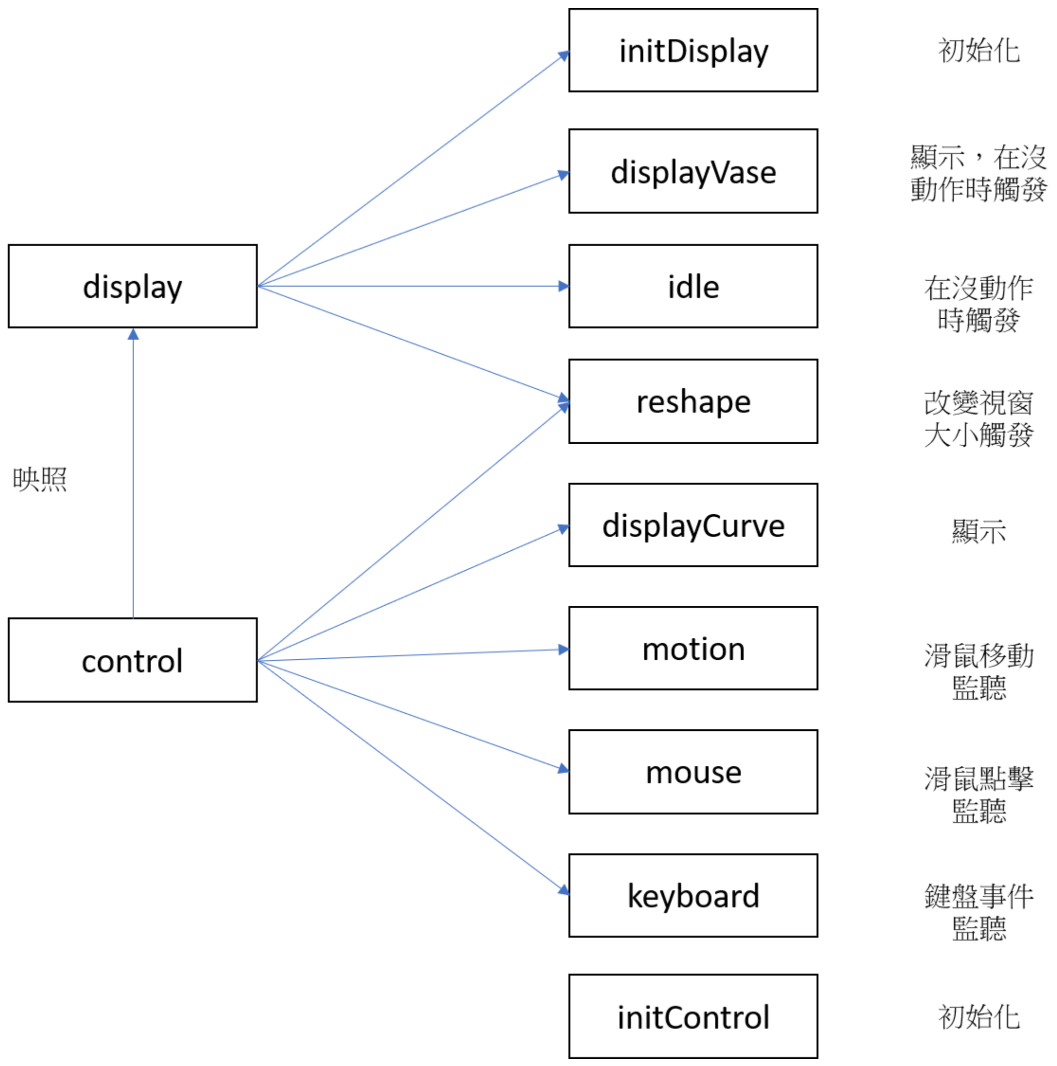
406261597 資工三甲 林子傑

參數化線性軸曲面設計

架構圖

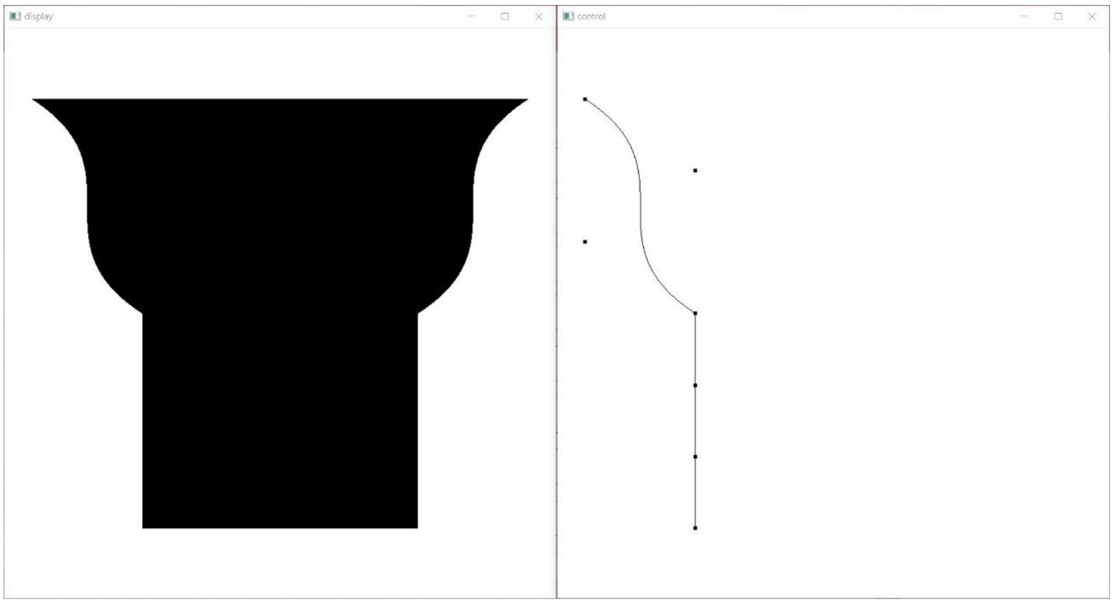


討論

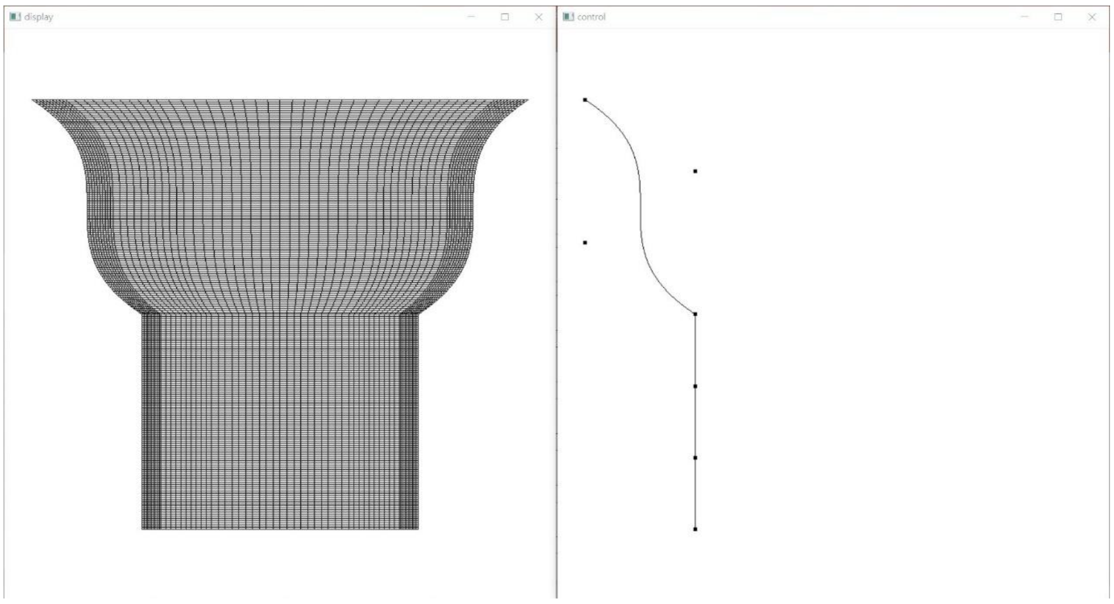
這份作業讓我知道程式利用視覺暫留的特性，將平面的圖形轉為立體，轉動不同的角度會有不同的效果，也讓我熟悉 openGL 的架構，keyboard 和 mouse 是如何觸發的，還有線性代數在電腦圖學的重要性，往後的作品我應該能更得心應手。

執行畫面

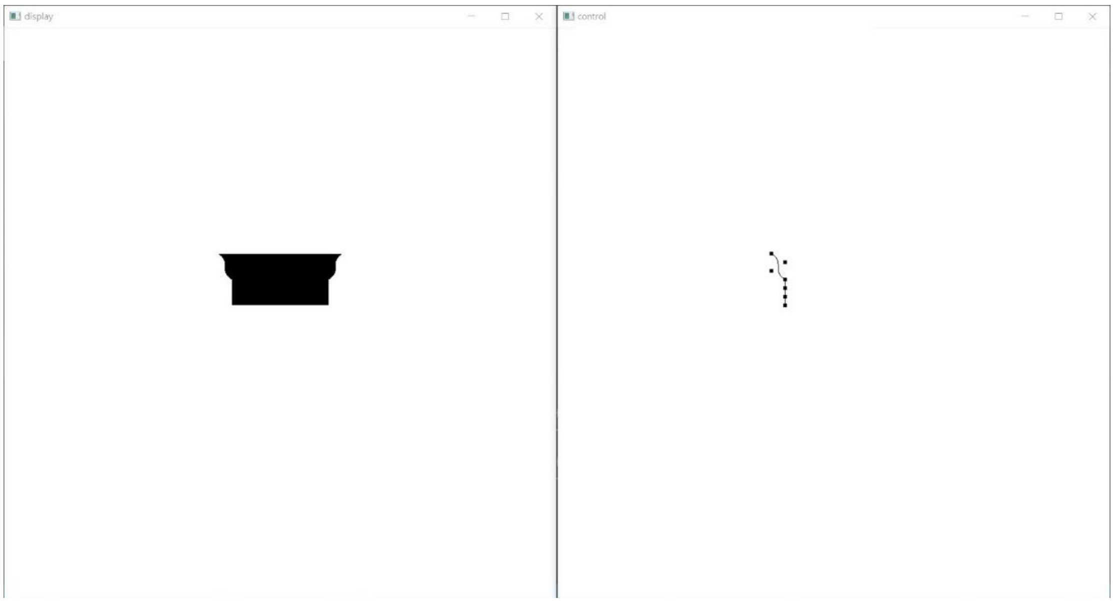
1. 塗色式



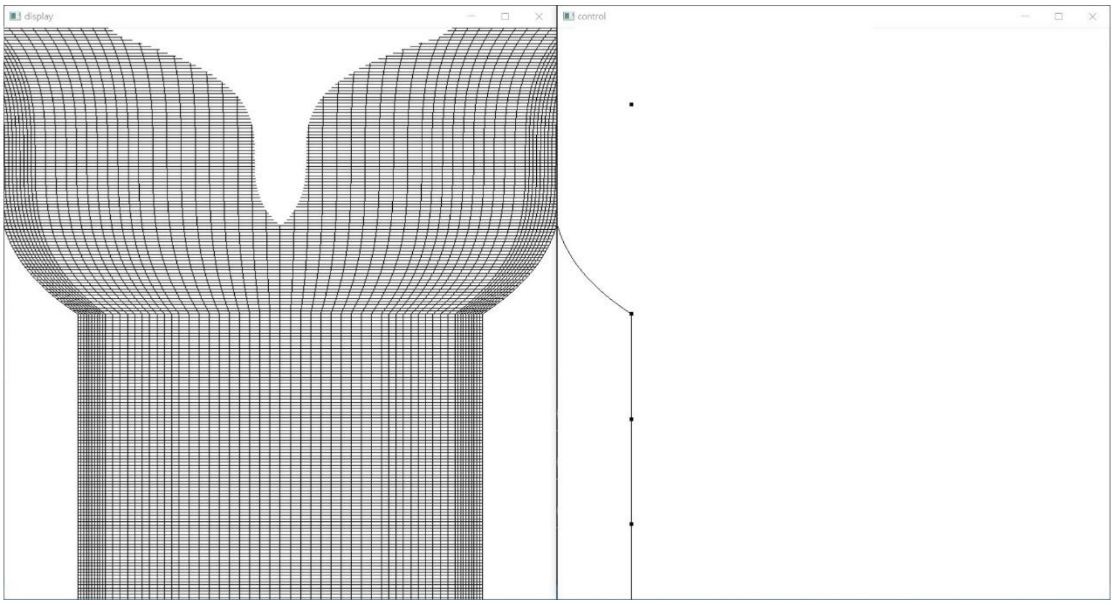
2.線框式



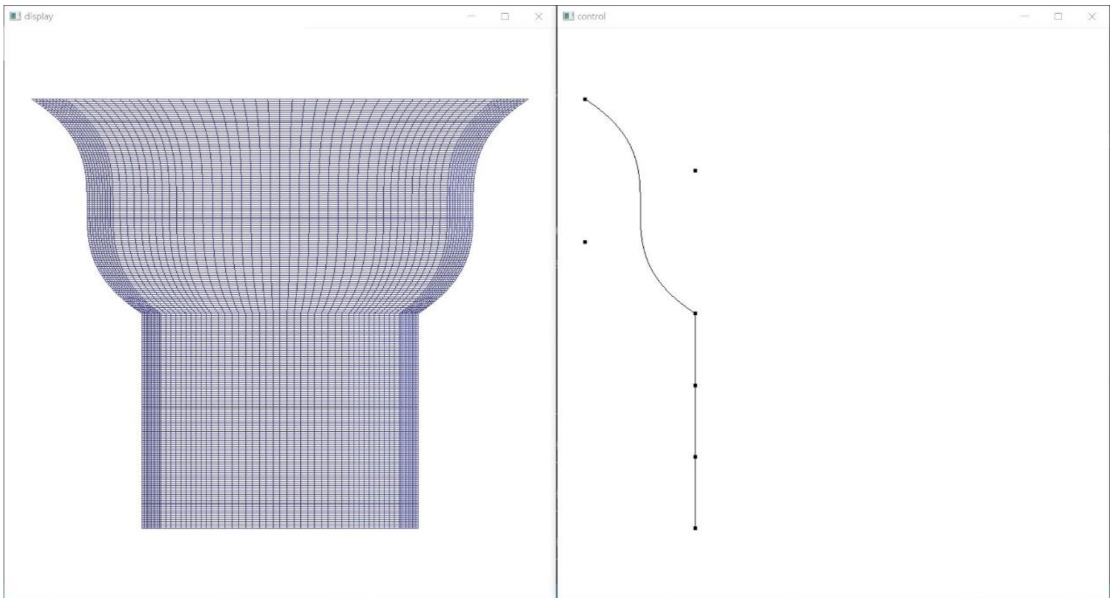
3. 縮小



4. 放大



5. 調色



程式碼

```
#ifdef __APPLE__
#include <GLUT/glut.h>

#else
#include <GL/glut.h>
#endif

#include <stdlib.h>
#include <stdio.h>
#include <cmath>
#include <ctime>
#define PI acos(-1)

int width = 400, height = 400;
int displayWindow, controlWindow;
const int POINT_NUM = 7;
int movePoint = -1;
int neonLight = 0;
GLenum style = GL_LINE;

float cpts[2][POINT_NUM][3];
float points[POINT_NUM][2] =
{
	{20, 50},
	{100, 100},
	{20, 150},
	{100, 200},
	{100, 250},
	{100, 300},
	{100, 350},
};
GLfloat rotatey[3][3]=
{
	// 30 degree
	{float(cos(PI/6.0)), 0.0, float(sin(PI/6.0))},
	{0.0, 1.0, 0.0},
	{float(-sin(PI/6.0)), 0.0, float(cos(PI/6.0))},
};

float color[1][3]={
        {0.0, 0.0, 0.0}
};

const GLfloat light_ambient[]    = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_diffuse[]    = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_position[] = { 1.0f, 1.0f, 0.0f, 1.0f };

const GLfloat mat_ambient[]        = { 0.8f, 0.8f, 0.8f, 1.0f };
const GLfloat mat_diffuse[]        = { 0.8f, 0.8f, 0.8f, 1.0f };
const GLfloat mat_specular[]      = { 0.8f, 0.8f, 0.8f, 1.0f };
const GLfloat high_shininess[] = { 100.0f };

void transCordinate(float x, float y, float &wx, float & wy)
{
        wx = (2.0 * x) / (float)(width - 1) - 1.0;
        wy = (2.0 * (height - 1 - y)) / (float)(height - 1) - 1.0;
}

void transMatrix(float a[3][3], float b[3], float c[3])
{
        for(int i = 0; i < 3; ++i)
        {
                c[i] = 0.0;
                for(int j = 0; j < 3; ++j)
                {
                        c[i] += a[i][j] * b[j];
                }
        }
}

void initControlPoints()
{
        float wx, wy;
        for(int i = 0; i < POINT_NUM; ++i)
```

```
	{
		transCordinate(points[i][0], points[i][1], wx, wy);
		cpts[0][i][0] = wx;
		cpts[0][i][1] = wy;
		cpts[0][i][2] = 0.0;
		transMatrix(rotatey, cpts[0][i], cpts[1][i]);
	}
}

static void matrix_cpy(GLfloat a[2][4][3], GLfloat b[2][7][3], int st)
{
	for(int i = 0; i < 2; ++i)
	{
		for(int j = 0; j < 4; ++j)
		{
			for(int k = 0; k < 3; ++k)
			{
				a[i][j][k] = b[i][j + st][k];
			}
		}
	}
}

static void drawCurves(){
	glColor3f(0.0, 0.0, 0.0);
	for(int i = 0; i + 3 < POINT_NUM; i += 3)
	{
		glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4, &cpts[0][i][0]);
		glMapGrid1f(100, 0.0, 1.0);
		glEvalMesh1(GL_LINE, 0, 100);
	}
}

static void drawVase()
{
	float tmp[2][4][3];
	glColor3fv(color[0]);
	for(int i = 0; i + 3 < POINT_NUM; i += 3)
	{
		matrix_cpy(tmp, cpts, i);
		glMap2f(GL_MAP2_VERTEX_3, 0.0, 1.0, 3, 4, 0.0, 1.0, 12, 2, &tmp[0][0][0]);
		glMapGrid2f(100, 0.0, 1.0, 10, 0.0, 1.0);
		glEvalMesh2(style, 0, 100, 0, 10);
	}
}

static void displayCurve()
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glColor3f(0.0, 0.0, 0.0);
	glutSetWindow(controlWindow);
	glutPostRedisplay();
	glPointSize(5.0);
	glBegin(GL_POINTS);
	for (int i = 0; i < POINT_NUM; i++)
		glVertex3fv(cpts[0][i]);
	glEnd();
	drawCurves();
	glutSwapBuffers();
}

static void displayVase()
{
	glutSetWindow(displayWindow);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glColor3f(0.0, 0.0, 0.0);
	for(int i = 0; i < 12; ++i){
		glRotated(30.0, 0.0, 1.0, 0.0);//rotate by y-axis
		drawVase();
	}
	glutSwapBuffers();
}

int nearestPoint(int x, int y)
{
	for(int i = 0; i < POINT_NUM; ++i)
	{
		if(abs(points[i][0] - x) <= 5 && abs(points[i][1] - y) <= 5)
		{
			return i;
		}
	}
	return -1;
}

void initDisplay()
{
	glClearColor(1.0, 1.0, 1.0, 1.0);
	initControlPoints();

	glEnable(GL_MAP2_VERTEX_3);
	glEnable(GL_DEPTH_TEST);
	glDepthFunc(GL_LESS);

	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);
	glEnable(GL_AUTO_NORMAL);
	glEnable(GL_COLOR_MATERIAL);

	glLightfv(GL_LIGHT0, GL_AMBIENT,    light_ambient);
	glLightfv(GL_LIGHT0, GL_DIFFUSE,    light_diffuse);
	glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
	glLightfv(GL_LIGHT0, GL_POSITION, light_position);


	glMaterialfv(GL_FRONT, GL_AMBIENT,      mat_ambient);
	glMaterialfv(GL_FRONT, GL_DIFFUSE,      mat_diffuse);
	glMaterialfv(GL_FRONT, GL_SPECULAR,    mat_specular);
	glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
}

void initControl()
{
	glClearColor(1.0, 1.0, 1.0, 1.0);
	glEnable(GL_MAP1_VERTEX_3);
}

static void reshape(int w, int h) {
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
	glMatrixMode(GL_MODELVIEW);
	glViewport(0, 0, w, h);
	width = w;
	height = h;
}

static void motion(int x, int y)
{
	if(movePoint != -1)
	{
		points[movePoint][0] = x;
		points[movePoint][1] = y;
		initControlPoints();
	}
}

static void mouse(int button, int state, int x, int y){
	if (button != GLUT_LEFT_BUTTON || state != GLUT_DOWN)
		return;
	if (button == GLUT_LEFT_BUTTON){
		movePoint = nearestPoint(x, y);
	}
}

static void randomColor()
{
	color[0][0] = float(rand() % 11) * 0.1;
	color[0][1] = float(rand() % 11) * 0.1;
	color[0][2] = float(rand() % 11) * 0.1;
}
```

```
void toLittle(float &tar, float base)
{
	if(tar > base / 2)
	{
		tar -= (tar - base / 2) * 0.1;
	}
	else
	{
		tar += (base / 2 - tar) * 0.1;
	}
}


void toLarger(float &tar, float base)
{
	if(tar > base / 2)
	{
		tar += (tar - base / 2) * 0.1;
	}
	else
	{
		tar -= (base / 2 - tar) * 0.1;
	}
}

static void keyBoard(unsigned char key, int x, int y)
{
	switch(key)
	{
		case 'q': case 'Q':
			exit(0);
			break;
		case 't': case 'T':
			style = (style == GL_LINE ? GL_FILL: GL_LINE);
			break;
		case 'w': case 'W':
			glutSetWindow(displayWindow);
			glRotated(-1.0, 1.0, 0.0, 0.0);
```

```
        break;
case 's': case 'S':
        glutSetWindow(displayWindow);
        glRotated(1.0, 1.0, 0.0, 0.0);
        break;
case 'a': case 'A':
        glutSetWindow(displayWindow);
        glRotated(-1.0, 0.0, 0.0, 1.0);
        break;
case 'd': case 'D':
        glutSetWindow(displayWindow);
        glRotated(1.0, 0.0, 0.0, 1.0);
        break;
case 'i': case 'I':
        color[0][0] = fmin(1.0, color[0][0] + 0.1);
        break;
case 'k': case 'K':
        color[0][0] = fmax(0.0, color[0][0] - 0.1);
        break;
case 'o': case 'O':
        color[0][1] = fmin(1.0, color[0][1] + 0.1);
        break;
case 'l': case 'L':
        color[0][1] = fmax(0.0, color[0][1] - 0.1);
        break;
case 'p': case 'P':
        color[0][2] = fmin(1.0, color[0][2] + 0.1);
        break;
case ';': case ':':
        color[0][2] = fmax(0.0, color[0][2] - 0.1);
        break;
case '[': case '{':
        neonLight = 0;
        break;
case ']': case '}':
        neonLight = 1;
        break;
case 'v': case 'V':
```

```
                for(int i = 0; i != POINT_NUM; ++i)
                {
                        toLittle(points[i][0], width);
                        toLittle(points[i][1], height);
                        initControlPoints();
                }
                break;
        case 'b': case 'B':
                for(int i = 0; i != POINT_NUM; ++i)
                {
                        toLarger(points[i][0], width);
                        toLarger(points[i][1], height);
                        initControlPoints();
                }
                break;
        }
}

static void idle()
{
        glutSetWindow(displayWindow);
        if(neonLight)
        {
                randomColor();
        }

        glutPostRedisplay();
}

int main(int argc, char *argv[])
{
        srand(time(NULL));
        glutInit(&argc, argv);
        glutInitWindowSize(width, height);

        glutInitWindowPosition(10, 10);
        glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
        initControlPoints();

        displayWindow = glutCreateWindow("display");
        initDisplay();
        glutDisplayFunc(displayVase);
        glutReshapeFunc(reshape);
        glutIdleFunc(displayVase);
        glutIdleFunc(idle);


        glutInitWindowPosition(800, 10);

        controlWindow = glutCreateWindow("control");
        initControl();
        glutDisplayFunc(displayCurve);
        glutReshapeFunc(reshape);
        glutMotionFunc(motion);
        glutMouseFunc(mouse);
        glutKeyboardFunc(keyBoard);

        glutMainLoop();

        return EXIT_SUCCESS;
}
```