期中考作業-參數化線性軸曲面設計

405262180

劉育騏

資工三乙

程式架構:

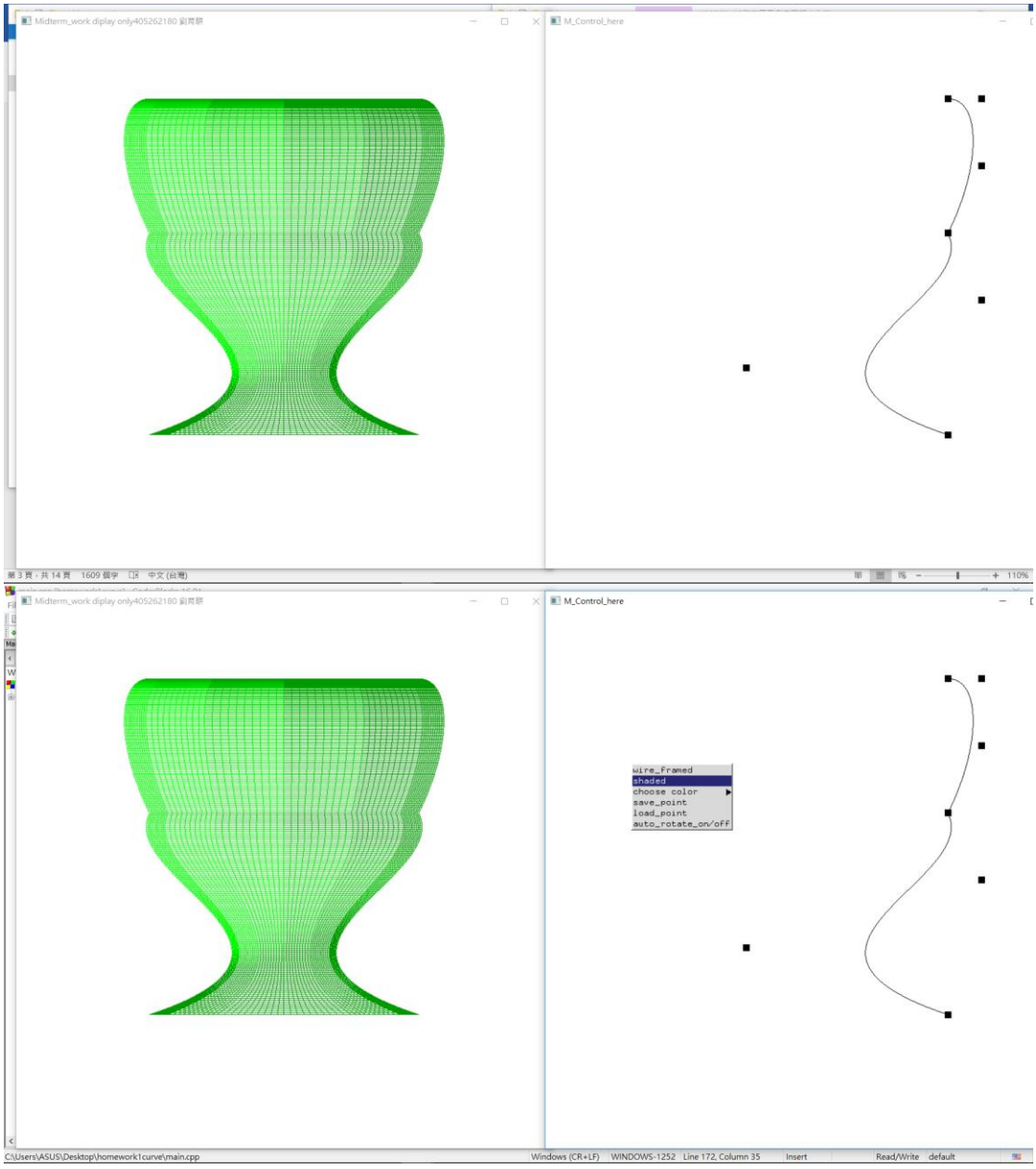
先給定 7 個點當第一組，對 y 旋轉 30 度後的 7 個點當第二組，使用第一組點切割成 0~3,3~6 使用 opengl 內建函式 map1f,grid1f,mesh1f 自動幫我做出 2 個曲線，在用第一組跟第二組點用二維求值器 map2f,grid2f,mesh2f 做出 2 個曲面後，重複對 y 軸旋轉 12 次 30 度印出來。得到一個本次作業所需要的結果。搬移控制點使用的是抓取滑鼠座標，並且在距離點小於等於 5 的情況下都算抓到點(點的大小是 10)，移動使用內建函式滑鼠 motion 更新座標，轉換我的座標系就能達到搬移控制點。完成上述要求。還有雙視窗能達到對哪個視窗旋轉而不互相干擾，所以我才用雙視窗。其他功能鍵盤:WASD 控制上下左右其實就是對 xz 旋轉就能達到，至於線框式跟塗滿式差別在於參數下的不同 GL_LINE 畫線框 GL_FILL 塗滿，子選單顏色來源:投影片範例，至於儲存與讀取座標其實就是多存一個陣列去記位置，本來想要用 stack 達到復原效果但礙於不能用 glut 好像不能用 c++，所以就沒去實作，然而自動旋轉就是用閒置的時候才旋轉，用變數去開跟關。

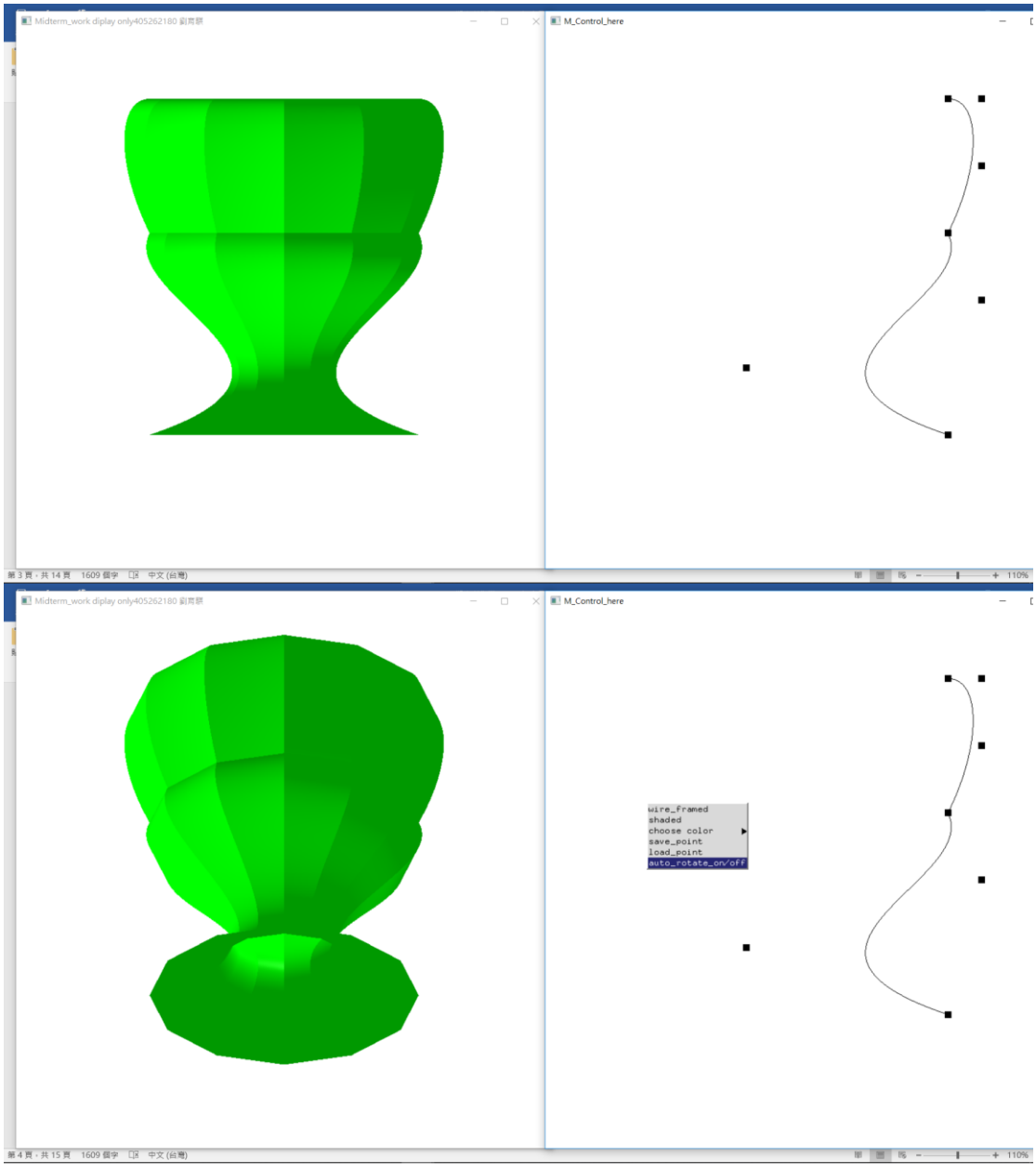
討論:

學到很多東西，以及用到不少好用的函式，只是不太清楚隱藏去除面之意思，因為看範例的 exe 感覺就有去除隱藏面了，感覺預設都

有去除。然而光源這個部分我還不熟悉，所以顏色上色起來，非常簡陋。光源材質部分還要在熟悉一下。然後也知道怎麼樣轉換坐標系。獲益匪淺。

執行畫面:





程式碼:

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

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

```
int width = 800, height = 800;
int main_w, sub_w;
const int MX = 7;
GLfloat cpts[2][MX][3];
GLenum chang = GL_LINE;
int pos[MX][2] = {
	{600, 600},
	{300, 500},
	{650, 400},
	{600, 300},
	{650, 200},
	{650, 100},
	{600, 100}
};
int sav[MX][2] = {
	{600, 600},
	{300, 500},
	{650, 400},
	{600, 300},
	{650, 200},
	{650, 100},
	{600, 100}
};
GLfloat rotatey[4][4]={//30degree
	{cos(PI/6.0), 0.0, sin(PI/6.0), 0.0},
	{0.0, 1.0, 0.0, 0.0},
	{-sin(PI/6.0), 0.0, cos(PI/6.0), 0.0},
	{0.0, 0.0, 0.0, 1.0},
};
int id = -1, cid = 8, on = 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 };
GLfloat colors[13][3]={{0.0, 0.0, 0.0}, {1.0, 0.0, 0.0},{0.0, 1.0, 0.0},
	{0.0, 0.0, 1.0}, {0.0, 1.0, 1.0}, {1.0, 0.0, 1.0}, {1.0, 1.0, 0.0},
	{0.5, 0.0, 0.0},{0.0, 0.5, 0.0},
	{0.0, 0.0, 0.5}, {0.0, 0.5, 0.5}, {0.5, 0.0, 0.5}, {0.5, 0.5, 0.0},
};
static void reshape(int, int);
static void drawCurves();
static void mouse(int, int, int, int);
int pick_point(int, int);
static void motion(int, int);
static void keyBoard(unsigned char, int, int);
void myinit();
static void draw3D();
static void matrix_mult(GLfloat m[4][4], GLfloat t[3], GLfloat r[3]);
static void display2();
static void main_menu(int index);
static void color_menu(int index);
static void set_matrix(GLfloat a[2][4][3], GLfloat b[2][7][3], int st);


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 set_matrix(GLfloat a[2][4][3], GLfloat b[2][7][3], int st){
	int i, j, k, l;
	for(i = 0; i < 2; ++i)
		for(j = 0, l = st; j < 4; ++j, ++l)
			for(k = 0; k < 3; ++k)
				a[i][j][k] = b[i][l][k];
```

```
}

static void drawCurves(){
	int i;
	glColor3f(0.0, 0.0, 0.0);
	for(i = 0; i+3 < MX; i+=3){
		glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4,
&cpts[0][i][0]);//define value to map1
		glMapGrid1f(100, 0.0, 1.0);//t=0~1.0
		glEvalMesh1(GL_LINE, 0, 100);//0~100 point->line
	}
}

static void display(void)
{
	glutSetWindow(main_w);

	int i;
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glColor3f(0.0, 0.0, 0.0);
	for(i = 0; i < 12; ++i){
		glRotated(30.0, 0.0, 1.0, 0.0);
		draw3D();
	}
	glutSwapBuffers();
}

static void display2(){

	int i;
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glColor3f(0.0, 0.0, 0.0);
	glutSetWindow(sub_w);
	glutPostRedisplay();
	glPointSize(10.0);
	glBegin(GL_POINTS);
	for (i = 0; i < MX; i++)
		glVertex3fv(cpts[0][i]);
```

```
	glEnd();
	drawCurves();

	glutSwapBuffers();
}

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){
		id = pick_point(x, y);
	}
}

int pick_point(int x, int y){
	int i;
	for(i = 0; i < MX; ++i){
		if( x >= pos[i][0] - 5 && x <= pos[i][0]+ 5 && y <= pos[i][1]+ 5
&& y >= pos[i][1] - 5)
			return i;
	}
	return -1;
}

static void motion(int x, int y){
	float wx, wy;
	int i;
	/* Translate back to our coordinate system */
	wx = (2.0 * x) / (float)(width - 1) - 1.0;
	wy = (2.0 * (height - 1 - y)) / (float)(height - 1) - 1.0;
	if(id==-1){
		return;
	}
	cpts[0][id][0] = wx;
	cpts[0][id][1] = wy;
	pos[id][0] = x;
	pos[id][1] = y;
	for(i = 0; i < MX; i++){//mov z next 7 point
```

```
            matrix_mult(rotatey, cpts[0][i], cpts[1][i]);
        }
    }

    static void keyBoard(unsigned char key, int x, int y){
        switch(key){
            case 27:
                exit(0);
            case 'w':case 'W':
                if(on)
                    on=0;
                glutSetWindow(main_w);
                glRotated(-30.0, 1.0, 0.0, 0.0);
                break;
            case 's':case 'S':
                if(on)
                    on=0;
                glutSetWindow(main_w);
                glRotated(30.0, 1.0, 0.0, 0.0);
                break;
            case 'a':case 'A':
                if(on)
                    on=0;
                glutSetWindow(main_w);
                glRotated(30.0, 0.0, 0.0, 1.0);
                break;
            case 'd':case 'D':
                if(on)
                    on=0;
                glutSetWindow(main_w);
                glRotated(-30.0, 0.0, 0.0, 1.0);
                break;
            }
    }

    static void draw3D(){
        int i;
        GLfloat tmp[2][4][3];
```

```
        glColor3fv(colors[cid]);
        for(i = 0; i + 3 < MX; i += 3){
        set_matrix(tmp, cpts, i);
        glMap2f(GL_MAP2_VERTEX_3, 0.0, 1.0, 3, 4, 0.0, 1.0, 12, 2,
&tmp[0][0][0]);//define value to map2
        glMapGrid2f(100, 0.0, 1.0, 10, 0.0, 1.0);//x_num of 100 r:0~1
y_num10 r:0~1
        glEvalMesh2(chang, 0, 100, 0, 10);//the same
        }

}

static void matrix_mult(GLfloat m[4][4], GLfloat t[3], GLfloat r[3]){
        int i, k;
        for(i = 0; i < 3; ++i){
                for(k = 0, r[i] = 0.0; k < 3; ++k){
                        r[i] +=     m[i][k] * t[k];
                }
        }
}

static void main_menu(int index){
        int i;
        switch(index){
                case 0:
                        chang = GL_LINE;
                break;
                case 1:
                        chang = GL_FILL;
                break;
                case 2:
                        for(i = 0; i < MX; ++i){
                                sav[i][0]=pos[i][0];
                                sav[i][1]=pos[i][1];
                        }
                        break;
                case 3:
                        float wx, wy;
```

```
                for(i = 0; i < MX; ++i){
                        pos[i][0]=sav[i][0];
                        pos[i][1]=sav[i][1];
                        wx = (2.0 * pos[i][0]) / (float)(width - 1) - 1.0;
                        wy = (2.0 * (height - 1 - pos[i][1])) / (float)(height - 1)
- 1.0;
                        cpts[0][i][0] = wx;
                        cpts[0][i][1] = wy;
                        cpts[0][i][2] = 0.0;
                        matrix_mult(rotatey, cpts[0][i], cpts[1][i]);
                }
                break;
        case 4:
                on^=1;
                break;
    }
}

static void color_menu(int index){
    switch(index){
        case 0:
                cid=0;
                break;
        case 1:
                cid=1;
                break;
        case 2:
                cid=2;
                break;
        case 3:
                cid=3;
                break;
        case 4:
                cid=4;
                break;
        case 5:
                cid=5;
                break;
```

```
            case 6:
                cid=6;
                break;
            case 7:
                cid=7;
                break;
            case 8:
                cid=8;
                break;
            case 9:
                cid=9;
                break;
            case 10:
                cid=10;
                break;
            case 11:
                cid=11;
                break;
            case 12:
                cid=12;
                break;
        }
}

void myinit(){
        glClearColor(1.0, 1.0, 1.0, 1.0);
        int i;
        float wx, wy;
        for(i = 0; i < MX; ++i){
            wx = (2.0 * pos[i][0]) / (float)(width - 1) - 1.0;
            wy = (2.0 * (height - 1 - pos[i][1])) / (float)(height - 1) - 1.0;
            cpts[0][i][0] = wx;
            cpts[0][i][1] = wy;
            cpts[0][i][2] = 0.0;
            matrix_mult(rotatey, cpts[0][i], cpts[1][i]);
        }

        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 myinit2(){
	glClearColor(1.0, 1.0, 1.0, 1.0);
	glEnable(GL_MAP1_VERTEX_3);//enable value

}

void idle(){
	glutSetWindow(main_w);
	if(on)
		glRotated(1.0, 1.0, 0.0, 0.0);
	glutPostRedisplay();
}

int main(int argc, char *argv[])
{
```

```
	int cm;
	glutInit(&argc, argv);
	glutInitWindowSize(width, height);
	glutInitWindowPosition(10, 10);
	glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);

	main_w = glutCreateWindow("Midterm_work diplay
only405262180 ¼B¨|ñ¯");
	myinit();
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutIdleFunc(display);
	glutIdleFunc(idle);

	glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
	glutInitWindowPosition(800, 10);
	sub_w = glutCreateWindow("M_Control_here");
	glutDisplayFunc(display2);
	myinit2();
	glutReshapeFunc(reshape);
	glutMotionFunc(motion);//move p
	glutMouseFunc(mouse);
	glutKeyboardFunc(keyBoard);

	cm = glutCreateMenu(color_menu);
	glutAddMenuEntry("Black",0);
	glutAddMenuEntry("Red",1);
	glutAddMenuEntry("Green",2);
	glutAddMenuEntry("Blue",3);
	glutAddMenuEntry("Cyan",4);
	glutAddMenuEntry("Magenta",5);
	glutAddMenuEntry("Yellow",6);
	glutAddMenuEntry("lowRed_good_use_on_shaded",7);
	glutAddMenuEntry("lowGreen_good_use_on_shaded",8);
	glutAddMenuEntry("lowBlue_good_use_on_shaded",9);
	glutAddMenuEntry("lowCyan_good_use_on_shaded",10);
	glutAddMenuEntry("lowMagenta_good_use_on_shaded",11);
	glutAddMenuEntry("lowYellow_good_use_on_shaded",12);
```

```
        glutCreateMenu(main_menu);
        glutAddMenuEntry("wire_framed",0);
        glutAddMenuEntry("shaded",1);
        glutAddSubMenu("choose color", cm);
        glutAddMenuEntry("save_point",2);
        glutAddMenuEntry("load_point",3);
        glutAddMenuEntry("auto_rotate_on/off",4);
        glutAttachMenu(GLUT_RIGHT_BUTTON);


        glutMainLoop();

        return EXIT_SUCCESS;
}
```