期中考作業-參數化線性軸曲面設計

405262180

劉育騏

資工三乙

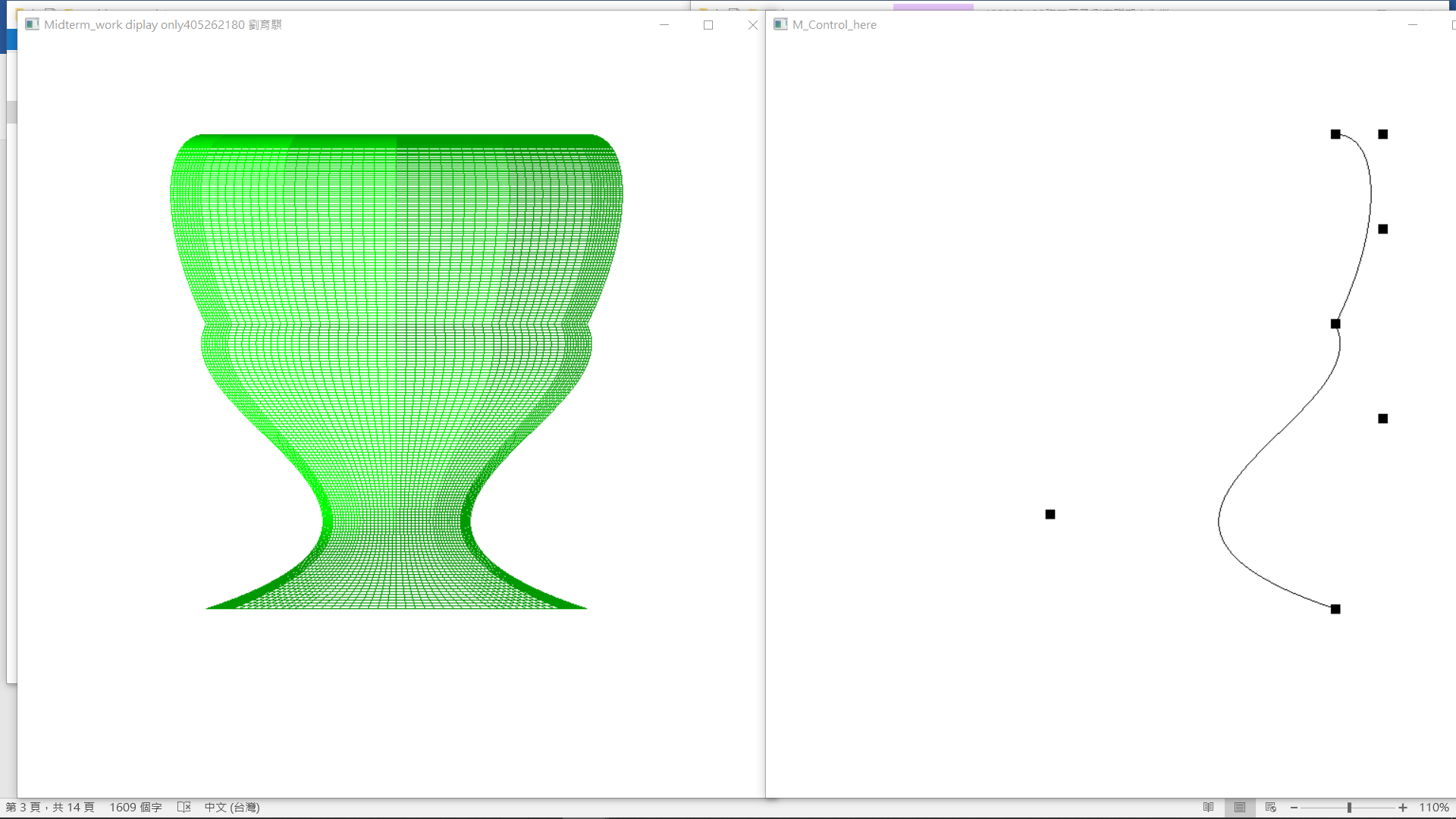
程式架構:

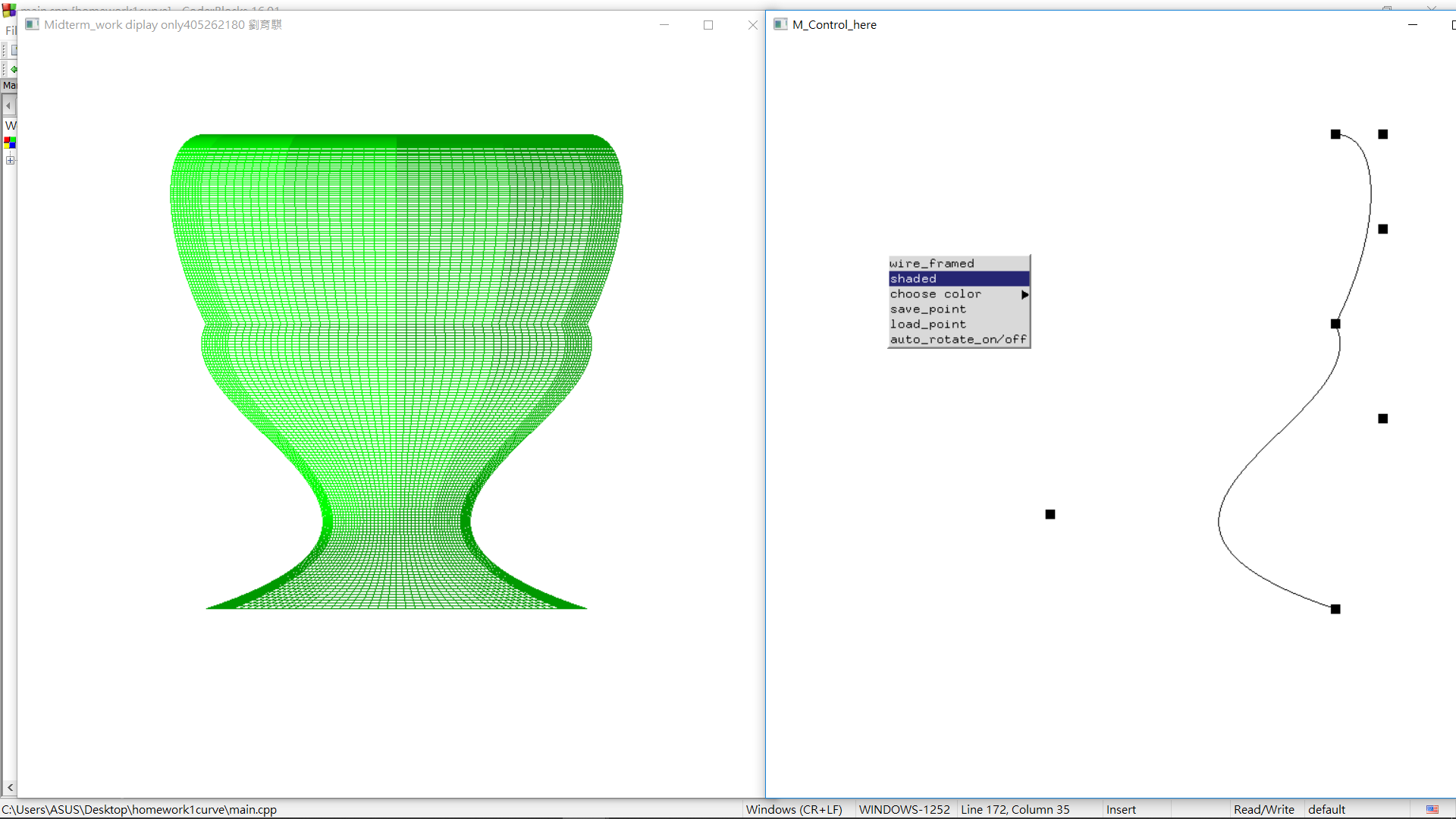
先給定7個點當第一組，對y旋轉30度後的7個點當第二組，使用第一組點切割成0~3,3~6使用opengl內建函式map1f,grid1f,mesh1f自動幫我做出2個曲線，在用第一組跟第二組點用二維求值器map2f,grid2f,mesh2f做出2個曲面後，重複對y軸旋轉12次30度印出來。得到一個本次作業所需要的結果。搬移控制點使用的是抓取滑鼠座標，並且在距離點小於等於5的情況下都算抓到點(點的大小是10)，移動使用內建函式滑鼠motion更新座標，轉換我的座標系就能達到搬移控制點。完成上述要求。還有雙視窗能達到對哪個視窗旋轉而不互相干擾，所以我才用雙視窗。其他功能鍵盤:WASD控制上下左右其實就是對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。

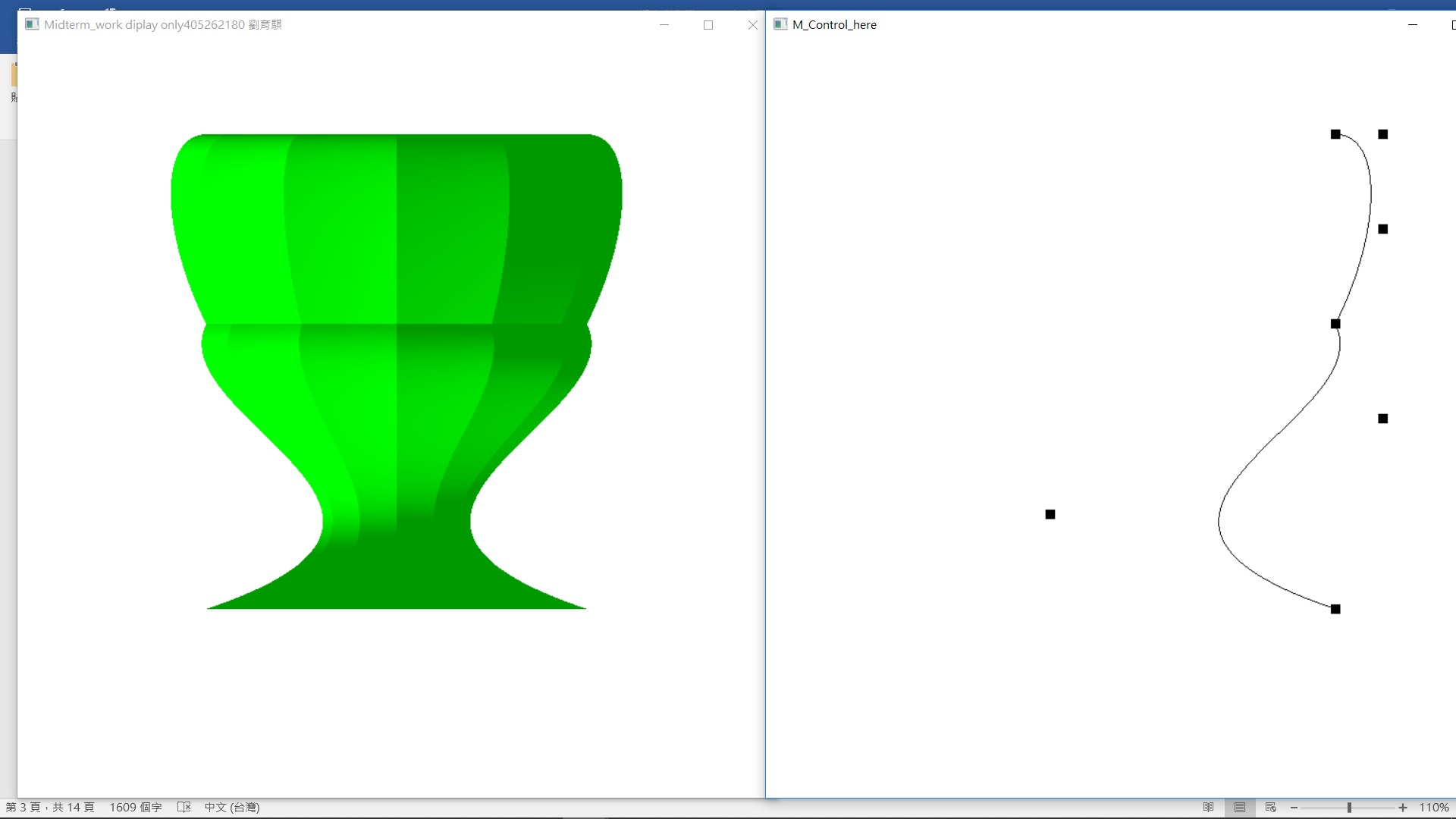
討論:

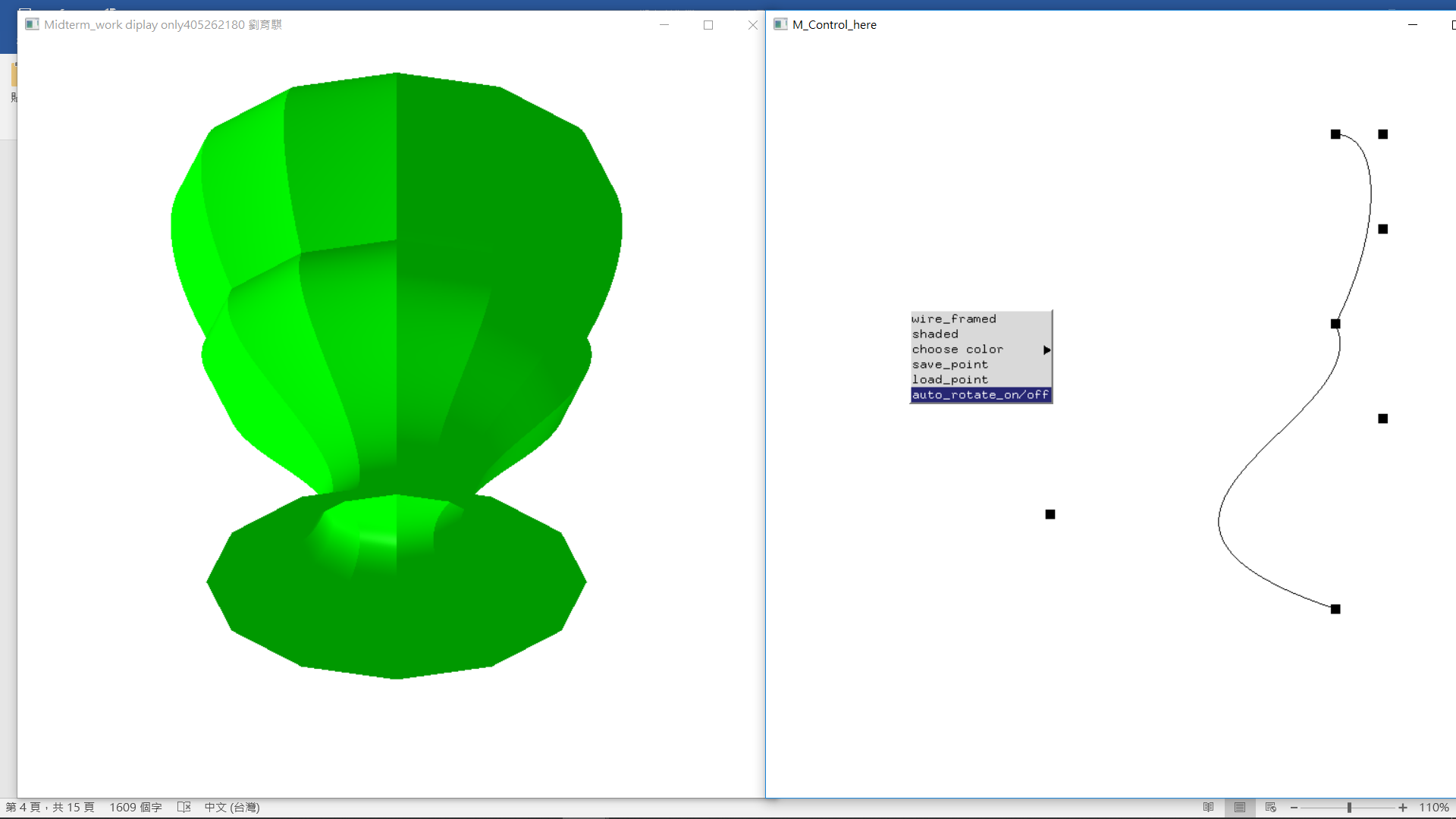
學到很多東西，以及用到不少好用的函式，只是不太清楚隱藏去除面之意思，因為看範例的exe感覺就有去除隱藏面了，感覺預設都有去除。然而光源這個部分我還不熟悉，所以顏色上色起來，非常簡陋。光源材質部分還要在熟悉一下。然後也知道怎麼樣轉換坐標系。獲益匪淺。

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;

}