

作業：

期中考替代作業(108 上)

參數化線性軸曲面設計

資工三乙-黃品翰 406262163

2019/12/20

## ● 程式架構

### - 左側視窗(Curve Drawer)

為平面貝茲曲線的呈現，分為上、下半部的貝茲曲線，各有兩個控制點與固定點，白線部分為七個點連線，固定點為青藍色，控制點則為綠色，選取時則會有紅色變化。白線為七個點連線，紅線則為貝茲曲線，渲染方式則是利用 `glMap1f`，劃出貝茲曲線。右鍵按下則有選單，可以選擇模式(WIRE, FILL, choose\_color, auto\_rotate, auto\_changecolor)

### - 右側視窗(Vase)

將左側視窗的貝茲曲線，將其視角繞  $y$  軸旋轉，則能呈現直線繞圈的效果。橫線繞圈則是在  $xz$  平面上，將原本二維的貝茲曲線的  $(x, y)$ ，轉換成三維空間的  $(x, y, z)$ ，畫圓圈(GL\_LINE\_STRIP)處理，塗色部分則用(GL\_QUAD\_STRIP)處理，上下左右按鍵則能調整視角

## ● 討論

### - 環境

利用 VScode 做文字編輯器，opengl 有專屬給 Mingw 的套件，將其 Lib 匯入則可以使用，這部分研究了一下子

### - 二維

二維圖形部分其實相較來說比較簡單，opengl 的 `glMap1f` 就能實作出來，一開始採到坑的部分應該是 `reshape` 及 `orthogonal`，網路上看了一下才大概，及比對學長姐們的 code 才知道在幹嘛

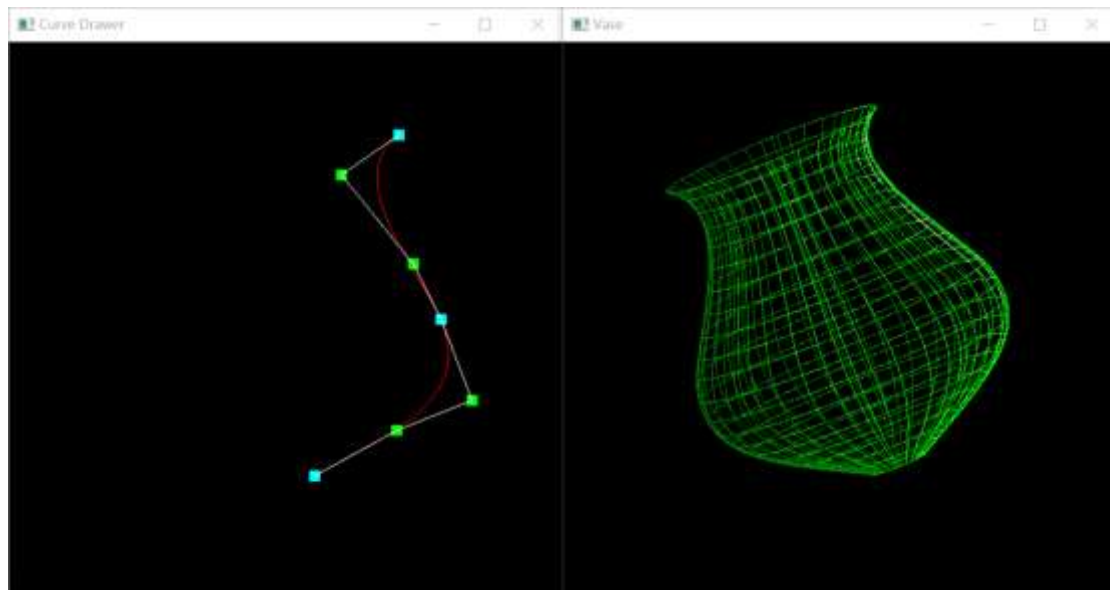
### - 三維及打光

一開始至實作出直線繞圈部分，橫線繞圓部分比較難去理解，因為需要把座標進行轉換，繞圓部分則是用圓的參數式來實作，利用多段直線來逼近圓，打光則是參考網路上的教學實作，不過跟預想的不太一樣，但因時間關係則沒有在另行調整

### - 開關(轉動、變色)

利用狀態機做處理，因為 opengl 的建構都是一直持續，因此要轉換狀態需用狀態機去實作，變色則只須改調色參數即可

## ● 執行畫面



## ● 程式碼

```
1. #include <math.h>
2. #include <stdlib.h>
3. #include <stdio.h>
4. #include <time.h>
5. #include <windows.h>
6. #include <limits>
7. #include <iostream>
8. #include <string>
9. #include "GL/glut.h"
10.
11. #define LINE_MODE 1
12. #define FILL_MODE 2
13. #define PI acos(-1)
14.
15. // 初始化設定
16. int windowHeight = 500;
17. int windowWidth = 500;
18. int numOfPoints = 7;
19. int selectedPoint = -1;
20. int cid = 0;
21. int mode = LINE_MODE;
```

```

22. float min[] = {-1.0, -1.0, -1.0};
23. float max[] = {1.0, 1.0, 1.0};
24. GLfloat light_ambient[] = {1.0f, 1.0f, 1.0f, 1.0f};
25. GLfloat light_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
26. GLfloat light_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
27. GLfloat light_position[] = {1.0f, 1.0f, 0.0f, 1.0f};
28. GLfloat mat_ambient[] = {0.8f, 0.8f, 0.8f, 1.0f};
29. GLfloat mat_diffuse[] = {0.8f, 0.8f, 0.8f, 1.0f};
30. GLfloat mat_specular[] = {0.8f, 0.8f, 0.8f, 1.0f};
31. GLfloat high_shininess[] = {100.0f};
32. GLfloat points[7][3] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
33. GLfloat rpoint[7][3] = {{140, 90, 0}, {300, 120, 0}, {365, 200, 0}, {390, 250, 0}, {380, 270, 0}, {350, 350, 0}, {250, 370, 0}};
34. float orthoMax[3], orthoMin[3];
35. float draw_point[21][2];
36.
37. //真實座標換成 opengl 平面視窗座標，利用 Orthographic projection
38. void real_point2ortho(int x, int y, GLfloat &fx, GLfloat &fy)
39. {
40.     fx = orthoMin[0] + (float)x / (float>windowWidth * (orthoMax[0] - orthoMin[0]);
41.     fy = orthoMin[1] + (float)(windowHeight - y) / (float>windowHeight * (orthoMax[1] - orthoMin[1]);
42. }
43.
44. // 算距離
45. GLfloat dist(GLfloat x1, GLfloat y1, GLfloat z1, GLfloat x2, GLfloat y2, GLfloat z2)
46. {
47.     int near_pt = -1;
48.     GLfloat d;
49.     d = (x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2) + (z1 - z2) * (z1 - z2);
50.     return d;
51. }
52. // 找離滑鼠最近的點
53. int check_near_point(GLfloat x, GLfloat y, GLfloat z)

```

```

54. {
55.     int idx = -1;
56.     GLfloat near_dis = std::numeric_limits<GLfloat>::max();
57.     for (int i = 0; i < 7; i++)
58.     {
59.         if (near_dis > dist(points[i][0], points[i][1], points[i][2], x, y,
60.             z))
61.         {
62.             near_dis = dist(points[i][0], points[i][1], points[i][2], x, y,
63.                 z);
64.             idx = i;
65.         }
66.     }
67.     return idx;
68. }
69. // mouse position
70. void motion(int x, int y)
71. {
72.     GLfloat fx, fy;
73.     real_point2ortho(x, y, fx, fy);
74.     // std::cout << x << " " << y << '\n';
75.     if (selectedPoint >= 0)
76.     {
77.         rpoint[selectedPoint][0] = x, rpoint[selectedPoint][1] = y;
78.         glutPostRedisplay();
79.     }
80. }
81. //mouse state
82. void Mouse(int button, int state, int x, int y)
83. {
84.     GLfloat fx, fy;
85.     real_point2ortho(x, y, fx, fy);
86.     if (button == GLUT_LEFT_BUTTON)
87.     {
88.         if (state == GLUT_DOWN)
89.         {
90.             selectedPoint = check_near_point(fx, fy, 0.0);

```

```

90.         glutMotionFunc(motion);
91.     }
92.     else
93.         selectedPoint = -1;
94. }
95. glutPostRedisplay();
96. glutSwapBuffers();
97. }
98.
99. //畫控制點
100. void draw_points()
101. {
102.     glPointSize(10.0f);
103.     glBegin(GL_POINTS);
104.     {
105.         for (int i = 0; i < numOfPoints; i++){
106.             if (i == 0 || i == 3 || i == 6)
107.             {
108.                 glColor3f(0.0, 1.0, 1.0);
109.                 glVertex3f(points[i][0], points[i][1], points[i][2]);
110.             }
111.             else{
112.                 glColor3f(0.0, 1.0, 0.0);
113.                 glVertex3f(points[i][0], points[i][1], points[i][2]);
114.             }
115.             if (i == selectedPoint)
116.             {
117.                 glColor3f(1.0, 0.0, 0.0);
118.                 glVertex3f(points[i][0], points[i][1], points[i][2]);
119.             }
120.         }
121.     }
122.     glEnd();
123.     // 將七個點連線，利用 GL_LINE_STRIP
124.     glColor3f(1.0, 1.0, 1.0);
125.     glBegin(GL_LINE_STRIP);
126.     {
127.         for (int i = 0; i < numOfPoints; i++)

```

```
128.         glVertex3f(points[i][0], points[i][1], points[i][2]);
129.     }
130.     glEnd();
131. }
132.
133.
134. void Reshape3D(int width, int height)
135. {
136.     // Find the largest and smallest values for all coordinates
137.     float max3D = 1.0f, min3D = -1.0f;
138.
139.     GLfloat aspect;
140.     windowWidth = width, windowHeight = height;
141.
142.     // Set the viewport
143.     // 把視景體截取的圖像按照怎樣的高和寬顯示到 screen
144.     glViewport(0, 0, (GLsizei)width, (GLsizei)height);
145.     // Make the projection matrix current
146.     glMatrixMode(GL_PROJECTION);
147.     // Clear the projection matrix
148.     glLoadIdentity();
149.
150.     // Set the projection matrix (based on the aspect ratio)
151.     // 因應視窗的大小變化，需要做長寬調整
152.     if (width <= height)
153.     {
154.         //y 座標需要被拉長
155.         aspect = (GLfloat)height / (GLfloat)width;
156.         orthoMin[0] = min3D;
157.         orthoMin[1] = min3D * aspect;
158.         orthoMin[2] = min3D;
159.         orthoMax[0] = max3D;
160.         orthoMax[1] = max3D * aspect;
161.         orthoMax[2] = max3D;
162.     }
163.     else
164.     {
165.         aspect = (GLfloat)width / (GLfloat)height;
```



```

166.         // x 座標需要被拉長
167.         orthoMin[0] = min3D * aspect;
168.         orthoMin[1] = min3D;
169.         orthoMin[2] = min3D;
170.         orthoMax[0] = max3D * aspect;
171.         orthoMax[1] = max3D;
172.         orthoMax[2] = max3D;
173.     }
174.
175.     //glOrtho(left,right,up,down,near,far)
176.     //利用 Orthographic projection
177.     //將立體座標壓成平面
178.     glOrtho(orthoMin[0], orthoMax[0],
179.             orthoMin[1], orthoMax[1],
180.             orthoMin[2], orthoMax[2]);
181.
182.     // Make the Model-View matrix active
183.     glMatrixMode(GL_MODELVIEW);
184. }
185.
186. void bezier_curve()
187. {
188.     glClear(GL_COLOR_BUFFER_BIT);
189.     //x = f(u); y = g(u); z = h(u);
190.     //將 7 個點換成 ortho 座標
191.     glPushMatrix();
192.     for (int i = 0; i < 7; i++)
193.     {
194.         GLfloat fx, fy;
195.         real_point2ortho(rpoint[i][0], rpoint[i][1], fx, fy);
196.         points[i][0] = fx;
197.         points[i][1] = fy;
198.         // std::cout << fx << " " << fy << '\n';
199.     }
200.     glMap1f(GL_MAP1_VERTEX_3, //生成的數據類型
201.             0.0f,             //u 值的下界
202.             100.0f,           //u 值的上界

```

```

203.          3,          //每個頂點在數據中的間隔，每一個頂點資訊都有
           x,y,z，所以長度為 3
204.          4,          //控制點的個數
205.          &points[0][0]); //其他點指向該控制點的 pointer
206.
207.          //利用劃線方式將點連成貝茲曲線
208.          glColor3f(1.0, 0.0, 0.0);
209.          glBegin(GL_LINE_STRIP);
210.          for (int i = 0; i < 100; i++)
211.          {
212.              glEvalCoord1f((GLfloat)i);
213.          }
214.          glEnd();
215.
216.          glMap1f(GL_MAP1_VERTEX_3, //生成的數據類型
217.                 0.0f,          //下界
218.                 100.0f,        //上界
219.                 3,             //每個頂點在數據中的間隔，每一個頂點資訊都有
           x,y,z，所以長度為 3
220.                 4,             //控制點的個數
221.                 &points[3][0]); //其他點指向該控制點的 pointer
222.
223.          glBegin(GL_LINE_STRIP);
224.          for (int i = 0; i < 100; i++)
225.          {
226.              glEvalCoord1f((GLfloat)i);
227.          }
228.          glEnd();
229.          glPopMatrix();
230.
231.          //將 evalCoord 中的點全部連線
232.          glEnable(GL_MAP1_VERTEX_3);
233.
234.          //畫 control point
235.          draw_points();
236.          // for mouse
237.          glutMouseFunc(Mouse);
238.          glutSwapBuffers();

```

```

239.     glutPostRedisplay();
240.
241. }
242.
243. //在 xz 平面畫圓，以 x 為半徑繞 y 軸旋轉畫圓
244. void draw_circle(float x, float y, float z, float radius)
245. {
246.     // glColor3f(0, 1, 0);
247.     int sections = 100;
248.     GLfloat TWOPI = 2.0f * 3.14159f;
249.     glBegin(GL_LINE_STRIP);
250.     for (int count = 0; count <= sections; count++)
251.     {
252.         glVertex3f(x + radius * cos(count * TWOPI / sections), y, z + radius * sin(count * TWOPI / sections));
253.     }
254.     glEnd();
255. }
256.
257.
258.
259. //xz 平面的圓
260. void Horizontal_circle()
261. {
262.     //貝茲曲線三次公式： $P_0 * (1-t)^3 + 3 * P_1 * t(1-t)^2 + 3 * P_2 * t^2(1-t) + P_3 * t^3$ 
263.     GLfloat P0_X = points[0][0], P0_Y = points[0][1];
264.     GLfloat P1_X = points[1][0], P1_Y = points[1][1];
265.     GLfloat P2_X = points[2][0], P2_Y = points[2][1];
266.     GLfloat P3_X = points[3][0], P3_Y = points[3][1];
267.     GLfloat P4_X = points[4][0], P4_Y = points[4][1];
268.     GLfloat P5_X = points[5][0], P5_Y = points[5][1];
269.     GLfloat P6_X = points[6][0], P6_Y = points[6][1];
270.
271.     for (GLfloat t = 0; t <= 1.1; t += 0.1)
272.     {
273.         GLfloat x = P0_X * pow((1 - t), 3) + 3 * P1_X * t * pow((1 - t), 2) + 3 * P2_X * pow(t, 2) * (1 - t) + P3_X * pow(t, 3);

```

```

274.         GLfloat y = P0_Y * pow((1 - t), 3) + 3 * P1_Y * t * pow((1 - t), 2
           ) + 3 * P2_Y * pow(t, 2) * (1 - t) + P3_Y * pow(t, 3);
275.         draw_point[(int)(t * 10)][0] = x;
276.         draw_point[(int)(t * 10)][1] = y;
277.         if(mode == LINE_MODE)
278.             draw_circle(0, y, 0, x);
279.     }
280.
281.     for (GLfloat t = 0; t <= 1.1; t += 0.1)
282.     {
283.         GLfloat x = P3_X * pow((1 - t), 3) + 3 * P4_X * t * pow((1 - t), 2
           ) + 3 * P5_X * pow(t, 2) * (1 - t) + P6_X * pow(t, 3);
284.         GLfloat y = P3_Y * pow((1 - t), 3) + 3 * P4_Y * t * pow((1 - t), 2
           ) + 3 * P5_Y * pow(t, 2) * (1 - t) + P6_Y * pow(t, 3);
285.         draw_point[10+(int)(t * 10)][0] = x;
286.         draw_point[10+(int)(t * 10)][1] = y;
287.         if (mode == LINE_MODE)
288.             draw_circle(0, y, 0, x);
289.     }
290. }
291.
292. void draw_surface_color()
293. {
294.     int sections = 10;
295.     GLfloat TWOPI = 2.0f * 3.14159f;
296.     glBegin(GL_QUAD_STRIP);
297.     for (int i = 0; i < 20; i++)
298.     {
299.         for (int count = 0; count <= sections; count++)
300.         {
301.             glVertex3f(draw_point[i][0] * cos(count * TWOPI / sections), d
               raw_point[i][1], draw_point[i][0] * sin(count * TWOPI / sections));
302.             glVertex3f(draw_point[i + 1][0] * cos(count * TWOPI / sections
               ), draw_point[i + 1][1], draw_point[i + 1][0] * sin(count * TWOPI / sections
               ));
303.         }
304.     }
305.     glEnd();

```

```

306.
307. }
308.
309.
310.
311. int x_view = 0;
312. int y_view = 0;
313. int z_view = 0;
314. GLfloat color_r = 0;
315. GLfloat color_g = 0;
316. GLfloat color_b = 0;
317.
318. void KeyBoards(unsigned char key, int mx, int my)
319. {
320.     switch (key)
321.     {
322.         case 'l':
323.             mode = LINE_MODE;
324.             break;
325.         case 'L':
326.             mode = LINE_MODE;
327.             break;
328.         case 'o':
329.             mode = FILL_MODE;
330.             break;
331.         case 'O':
332.             mode = FILL_MODE;
333.             break;
334.     }
335. }
336. int save_view = 0;
337. void SpecialKey(GLint key, int mx, int my)
338. {
339.     if(x_view >= 360 || x_view <= -360)
340.         x_view = 0, save_view = x_view;
341.
342.     if (key == GLUT_KEY_UP)
343.     {

```

```
344.         x_view -= 1;
345.         save_view = x_view;
346.     }
347.     if (key == GLUT_KEY_LEFT)
348.     {
349.         z_view -= 1;
350.     }
351.     if (key == GLUT_KEY_DOWN)
352.     {
353.         x_view += 1;
354.         save_view = x_view;
355.     }
356.     if (key == GLUT_KEY_RIGHT)
357.     {
358.         z_view += 1;
359.     }
360. }
361.
362. void vase()
363. {
364.     // Initialize the Model-View matrix
365.     glMatrixMode(GL_MODELVIEW);
366.     glLoadIdentity();
367.
368.     // glPolygonMode 正反面都是用線或填滿
369.     if (mode == LINE_MODE)
370.     {
371.         glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
372.     }
373.     else
374.     {
375.         glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
376.     }
377.
378.     //在 x,z 平面，繞 y 軸旋轉
379.     glutKeyboardFunc(KeyBoards);
380.     glutSpecialFunc(SpecialKey);
381.     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```

382.
383.     int idx = 0;
384.
385.     if (mode == LINE_MODE)
386.         glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
387.     else
388.         glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
389.
390.     //隨 x_view/z_view 做視角調整
391.     glRotatef(x_view, 1, 0, 0);
392.     glRotatef(z_view, 0, 0, 1);
393.
394.     for (int i = 0; i < 360; i += 3)
395.     {
396.         glColor3f(color_r, color_g, color_b);
397.         glRotatef(y_view + i, 0, 1, 0);
398.         GLfloat fx, fy;
399.         for (int j = 0; j < 7; j++)
400.         {
401.             real_point2ortho(rpoint[j][0], rpoint[j][1], fx, fy);
402.             points[j][0] = fx;
403.             points[j][1] = fy;
404.         }
405.         if (mode == LINE_MODE)
406.             Horizontal_circle();
407.         else{
408.             Horizontal_circle();
409.             draw_surface_color();
410.         }
411.
412.
413.         glMap1f(GL_MAP1_VERTEX_3, //生成的數據類型
414.                0.0f,           //u 值的下界
415.                500.0f,         //u 值的上界
416.                3,              //每個頂點在數據中的間隔，每一個頂點資訊都有
                                //x,y,z，所以長度為 3
417.                4,              //控制點的個數
418.                &points[0][0]); //其他點指向該控制點的 pointer

```

```

419.
420.     //利用劃線方式將點連成貝茲曲線
421.     // glColor3f(0.0, 1.0, 0.0);
422.     glBegin(GL_LINE_STRIP);
423.     for (int i = 0; i < 500; i++)
424.         glEvalCoord1f((GLfloat)i);
425.     glEnd();
426.
427.     glMap1f(GL_MAP1_VERTEX_3, //生成的數據類型
428.             0.0f,             //下界
429.             500.0f,           //上界
430.             3,                //每個頂點在數據中的間隔，每一個頂點資訊都有
                                //x,y,z，所以長度為 3
431.             4,                //控制點的個數
432.             &points[3][0]);   //其他點指向該控制點的 pointer
433.
434.     glBegin(GL_LINE_STRIP);
435.     for (int i = 0; i < 500; i++)
436.         glEvalCoord1f((GLfloat)i);
437.     glEnd();
438.
439.     //將 evalCoord 中的點全部連線
440.     glEnable(GL_MAP1_VERTEX_3);
441. }
442.
443.     glutPostRedisplay();
444.     glutSwapBuffers();
445. }
446.
447.
448.
449.     bool auto_rotate = false;
450.     bool auto_change_color = false;
451.     int save_color = 6;
452.
453.     GLfloat colors[7][3] = {
454.         {0.0, 0.0, 0.0},
455.         {1.0, 0.0, 0.0},

```



```
456.     {0.0, 1.0, 0.0},
457.     {0.0, 0.0, 1.0},
458.     {0.0, 1.0, 1.0},
459.     {1.0, 0.0, 1.0},
460.     {1.0, 1.0, 0.0},
461. };
462.
463. void auto_color()
464. {
465.     if (auto_change_color)
466.     {
467.         if (save_color == 0)
468.             save_color = 1;
469.         else if (save_color == 1)
470.             save_color = 2;
471.         else if (save_color == 2)
472.             save_color = 3;
473.         else if (save_color == 3)
474.             save_color = 4;
475.         else if (save_color == 4)
476.             save_color = 5;
477.         else if (save_color == 5)
478.             save_color = 6;
479.         else if (save_color == 6)
480.             save_color = 0;
481.         color_r = colors[save_color][0];
482.         color_g = colors[save_color][1];
483.         color_b = colors[save_color][2];
484.     }
485.     else
486.     {
487.         color_r = colors[save_color][0];
488.         color_g = colors[save_color][1];
489.         color_b = colors[save_color][2];
490.     }
491. }
492.
493. void auto_rot()
```

```
494. {
495.     if (auto_rotate)
496.     {
497.         if(save_view >= 360 || save_view <= -360)
498.             x_view = 0;
499.             x_view += 1;
500.             save_view = x_view;
501.     }
502.     else
503.     {
504.         x_view = save_view;
505.     }
506. }
507.
508. int vase_window;
509.
510. void idle()
511. {
512.     glutSetWindow(vase_window);
513.     auto_color();
514.     auto_rot();
515. }
516.
517.
518. void main_menu(int index)
519. {
520.     switch (index)
521.     {
522.         case 0:
523.             mode = LINE_MODE;
524.             break;
525.         case 1:
526.             mode = FILL_MODE;
527.             break;
528.         case 2:
529.             break;
530.         case 3:
531.             auto_change_color ^= 1;
```

```

532.         break;
533.     case 4:
534.         auto_rotate ^= 1;
535.         break;
536.     }
537. }
538.
539. static void color_menu(int index)
540. {
541.     if ((index <= 6) && (index >= 0))
542.     {
543.         save_color = index;
544.     }
545. }
546.
547. //MENU
548. void menu()
549. {
550.     int cm = glutCreateMenu(color_menu);
551.     glutAddMenuEntry("Black", 0);
552.     glutAddMenuEntry("Red", 1);
553.     glutAddMenuEntry("Green", 2);
554.     glutAddMenuEntry("Blue", 3);
555.     glutAddMenuEntry("Cyan", 4);
556.     glutAddMenuEntry("Magenta", 5);
557.     glutAddMenuEntry("Yellow", 6);
558.     glutCreateMenu(main_menu);
559.     glutAddMenuEntry("WIRE", 0);
560.     glutAddMenuEntry("FILL", 1);
561.     glutAddSubMenu("choose_color", cm);
562.     glutAddMenuEntry("auto_rotate on/off", 4);
563.     glutAddMenuEntry("auto_change color on/off", 3);
564.     glutAttachMenu(GLUT_RIGHT_BUTTON);
565. }
566.
567. int main(int argc, char *argv[])
568. {
569.     //opengl 基礎設定

```

```
570.     glutInit(&argc, argv);
571.     glEnable(GL_MAP1_VERTEX_3);
572.     glEnable(GL_DEPTH_TEST);
573.     glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
574.
575.
576.     //視窗設定
577.     glutInitWindowPosition(100, 100); // 設定視窗位置
578.     glutInitWindowSize(windowWidth, windowHeight); // 設定視窗大小
579.
580.     //視窗顏色設定
581.     glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
582.     glColor3f(1.0f, 0.0f, 1.0f);
583.
584.     // render point/curve
585.     // reshape 為視窗更動時，圖形長寬會更著變動
586.     glutCreateWindow("Curve Drawer"); // 設定視窗標題
587.     glutDisplayFunc(bezier_curve);
588.     glutReshapeFunc(Reshape3D);
589.
590.     //MENU
591.     menu();
592.
593.
594.     // render vase
595.     // reshape 為視窗更動時，圖形長寬會更著變動
596.     glutInitWindowPosition(600, 100); // 設定視窗位置
597.     vase_window = glutCreateWindow("Vase"); // 設定視窗標題
598.     glutDisplayFunc(vase);
599.     glutReshapeFunc(Reshape3D);
600.     glutIdleFunc(vase);
601.     glutIdleFunc(idle);
602.
603.
604.     glEnable(GL_LIGHTING);
605.     glEnable(GL_LIGHT0);
606.     glEnable(GL_DEPTH_TEST);
607.     glEnable(GL_AUTO_NORMAL);
```

```
608.     glEnable(GL_COLOR_MATERIAL);
609.
610.     GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};
611.     GLfloat mat_shininess[] = {100.0};
612.     GLfloat light_position[] = {1.0, 1.0, 1.0, 1.0};
613.     GLfloat white_light[] = {1.0, 1.0, 1.0, 1.0};
614.     GLfloat Light_Model_Ambient[] = {0.5, 0.5, 1, 1};
615.
616.     glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
617.     glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
618.
619.     glLightfv(GL_LIGHT0, GL_POSITION, light_position);
620.     glLightfv(GL_LIGHT0, GL_DIFFUSE, white_light);
621.     glLightfv(GL_LIGHT0, GL_SPECULAR, white_light);           //镜面反
射光
622.     glLightModelfv(GL_LIGHT_MODEL_AMBIENT, Light_Model_Ambient); //环境光
参数
623.
624.
625.
626.     glutMainLoop();
627.     return 0;
628. }
```