**計算機圖學單元介紹**

一、英文主題：

Chapter 12: Curves and Surfaces System

二、中文主題

單元12：曲面描圖

三、組別：

第07組

四、組員：

B0829001\_姜念廷；B0829009\_王凱心；B0829020\_邵思絜；B0829021\_游婷婷

B0829032\_廖珮萱；B0829049\_朱紫綸：B0829059\_呂欣玲

五、作業分工：

(詳見作業報告)

六、功能簡述：

本單元介紹曲線與曲面的主要三種繪圖方式、技巧與其背後的數學原理，此三種繪圖方式分別為Interpolating, Bezier與 Spline curves，以及在繪製曲線、曲面時需要考慮的五個設計規則，分別為平滑性與連續性、形狀的局部控制、穩定性、容易渲染與容易計算導入的能力。

七、主要程式碼：

相關檔案：Ch\_12\_tm7\_src1.cpp

|  |
| --- |
| #include <GL/glut.h>  typedef GLfloat point[3];  point data[32][4][4];  /\* 306 vertices \*/  #include "vertices.h"  /\* 32 patches each defined by 16 vertices, arranged in a 4 x 4 array \*/  /\* NOTE: numbering scheme for teapot has vertices labeled from 1 to 306 \*/  /\* remnent of the days of FORTRAN \*/  #include "patches.h"  void display(void)  {  int i, j, k;  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  glColor3f(1.0, 1.0, 1.0);  glLoadIdentity();  glTranslatef(0.0, 0.0, -10.0);  glRotatef(-35.26, 1.0, 0.0, 0.0);  glRotatef(-45.0, 0.0, 1.0, 0.0);  /\* gluLookAt(-1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); \*/  /\* data aligned along z axis, rotate to align with y axis \*/  glRotatef(-90.0, 1.0,0.0, 0.0);  for(k=0;k<32;k++)  {  glMap2f(GL\_MAP2\_VERTEX\_3, 0, 1, 3, 4,  0, 1, 12, 4, &data[k][0][0][0]);  for (j = 0; j <= 8; j++)  {  glBegin(GL\_LINE\_STRIP);  for (i = 0; i <= 30; i++)  glEvalCoord2f((GLfloat)i/30.0, (GLfloat)j/8.0);  glEnd();  glBegin(GL\_LINE\_STRIP);  for (i = 0; i <= 30; i++)  glEvalCoord2f((GLfloat)j/8.0, (GLfloat)i/30.0);  glEnd();  }  }  glFlush();  }  void  myReshape(int w, int h)  {  glViewport(0, 0, w, h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  if (w <= h)  glOrtho(-4.0, 4.0, -4.0 \* (GLfloat) h / (GLfloat) w,  4.0 \* (GLfloat) h / (GLfloat) w, -20.0, 20.0);  else  glOrtho(-4.0 \* (GLfloat) w / (GLfloat) h,  4.0 \* (GLfloat) w / (GLfloat) h, -4.0, 4.0, -20.0, 20.0);  glMatrixMode(GL\_MODELVIEW);  display();  }  void myinit()  {  glEnable(GL\_MAP2\_VERTEX\_3);  glMapGrid2f(20, 0.0, 1.0, 20, 0.0, 1.0);  glEnable(GL\_DEPTH\_TEST);  }  main(int argc, char \*argv[])  {  int i,j, k, m, n;  for(i=0;i<32;i++) for(j=0;j<4;j++) for(k=0;k<4;k++) for(n=0;n<3;n++)  {  /\* put teapot data into single array for subdivision \*/  m=indices[i][j][k];  for(n=0;n<3;n++) data[i][j][k][n]=vertices[m-1][n];  }  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(500, 500);  glutCreateWindow("teapot");  myinit();  glutReshapeFunc(myReshape);  glutDisplayFunc(display);  glutMainLoop();  } |

八、程式說明：

1. typedef GLfloat point[3]：宣告一個 typedef name，point。
2. void display(void)：呼叫各個函式，繪製出茶壺
3. void myReshape()：當窗口尺寸改變時，圖形比例不發生變化
4. glMatrixMode(GL\_PROJECTION)：將當前矩陣指定為投影矩陣。
5. void glMap2f(GLenum target, GLfloat u1, GLfloat u2, GLint ustride, GLint uorder, GLfloat v1, GLfloat v2, GLint vstride, GLint vorder, const GLfloat \*points)：允許二維映射，

GLenum target：生成目標，最終生成繪製點用GL\_MAP2\_VERTEX\_3，GL\_MAP2\_VERTEX\_3為頂點模式。

GLfloat u1、u2：貝賽爾曲線模型的參數，這個參數定義域為[0,1]，u1代表下限，u2代表上限，一般分別設為0和1。

GLint ustride：u方向上一個控制點與下一個控制點的距離。

GLint uorder：曲線的階數，等於曲線上的控制點數量。

GLint vstride：v方向上一個控制點與下一個控制點的距離。

GLint vorder：曲線的階數，等於曲線上的控制點數量。

const GLfloat \*points：指向控制點數組的指針。

1. glViewport( GLint x, GLint y, GLsizei width, GLsizei height )：x、y 以像素為單位，指定了視窗的左下角位置。width、height 表示這個視窗矩形的寬度和高度，根據窗口的實時變化重繪窗口。
2. glOrtho(left, right, bottom, top, near, far)：建立一個正交平行的視景體。left表示視景體左面的座標;right表示右面的座標;bottom表示下面的座標;top表示上面的座標;near表示靠近深度裁剪平面的距離，如果平面位於檢視器後方，此距離就會是負數;far表示深度裁剪平面的距離，如果平面位於檢視器後方，此距離就會是負數。
3. glBegin(GL\_LINE\_STRIP)：繪製一系列連接的直線段，從起始點到終止點，不形成環路。
4. glEvalCoord2f()：使用兩個定義域值(u和v)來評估啟用的二維地圖。

九、延伸應用程式碼：

相關檔案：Ch\_12\_tm7\_src2.cpp

|  |
| --- |
| #include <stdlib.h>  #include <GL/glut.h>  typedef enum  {  BEZIER,  INTERPOLATED,  BSPLINE  } curveType;  void keyboard(unsigned char key, int x, int y);  void computeMatrix(curveType type, float m[4][4]);  void vmult(float m[4][4], float v[4][3], float r[4][3]);  /\* Colors to draw them in \*/  GLfloat colors[][3] =  {  { 1.0, 0.0, 0.0 },  { 0.0, 1.0, 0.0 },  { 0.0, 0.0, 1.0 }  };  #define MAX\_CPTS 25 /\* Fixed maximum number of control points \*/  GLfloat cpts[MAX\_CPTS][3];  int ncpts = 0;  static int width = 500, height = 500; /\* Window width and height \*/  /\* Matrix stuff \*/  /\* This routine multiplies two 4 x 4 matrices. \*/  /\* This routine multiplies a 4 x 4 matrix with a vector of 4 points. \*/  void vmult(float m[4][4], float v[4][3], float r[4][3])  {  int i, j, k;  for (i = 0; i < 4; i++)  for (j = 0; j < 3; j++)  for (k = 0, r[i][j] = 0.0; k < 4; k++)  r[i][j] += m[i][k] \* v[k][j];  }  /\* Interpolating to Bezier matrix \*/  static float minterp[4][4] =  {  { 1.0, 0.0, 0.0, 0.0 },  { -5.0/6.0, 3.0, -3.0/2.0, 1.0/3.0 },  { 1.0/3.0, -3.0/2.0, 3.0, -5.0/6.0 },  { 0.0, 0.0, 0.0, 1.0 },  };  /\* B-spline to Bezier matrix \*/  static float mbspline[4][4] =  {  { 1.0/6.0, 4.0/6.0, 1.0/6.0, 0.0 },  { 0.0, 4.0/6.0, 2.0/6.0, 0.0 },  { 0.0, 2.0/6.0, 4.0/6.0, 0.0 },  { 0.0, 1.0/6.0, 4.0/6.0, 1.0/6.0 },  };  static float midentity[4][4] =  {  { 1.0, 0.0, 0.0, 0.0},  { 0.0, 1.0, 0.0, 0.0},  { 0.0, 0.0, 1.0, 0.0},  { 0.0, 0.0, 0.0, 1.0}  };  /\* Calculate the matrix used to transform the control points \*/  void computeMatrix(curveType type, float m[4][4])  {  int i, j;  switch (type)  {  case BEZIER:  /\* Identity matrix \*/  for (i = 0; i < 4; i++)  for (j = 0; j < 4; j++)  m[i][j] = midentity[i][j];  break;  case INTERPOLATED:  for (i = 0; i < 4; i++)  for (j = 0; j < 4; j++)  m[i][j] = minterp[i][j];  break;  case BSPLINE:  for (i = 0; i < 4; i++)  for (j = 0; j < 4; j++)  m[i][j] = mbspline[i][j];  break;  }  }  /\* Draw the indicated curves using the current control points. \*/  static void drawCurves(curveType type)  {  int i;  int step;  GLfloat newcpts[4][3];  float m[4][4];  /\* Set the control point computation matrix and the step size. \*/  computeMatrix(type, m);  if(type == BSPLINE) step = 1;  else step = 3;  glColor3fv(colors[type]);  /\* Draw the curves \*/  i = 0;  while (i + 3 < ncpts)  {  /\* Calculate the appropriate control points \*/  vmult(m, &cpts[i], newcpts);  /\* Draw the curve using OpenGL evaluators \*/  glMap1f(GL\_MAP1\_VERTEX\_3, 0.0, 1.0, 3, 4, &newcpts[0][0]);  glMapGrid1f(30, 0.0, 1.0);  glEvalMesh1(GL\_LINE, 0, 30);  /\* Advance to the next segment \*/  i += step;  }  glFlush();  }  /\* This routine displays the control points \*/  static void display(void)  {  int i;  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3f(0.0, 0.0, 0.0);  glPointSize(5.0);  glBegin(GL\_POINTS);  for (i = 0; i < ncpts; i++)  glVertex3fv(cpts[i]);  glEnd();  glFlush();  }  /\* This routine inputs new control points \*/  static void mouse(int button, int state, int x, int y)  {  float wx, wy;  /\* We are only interested in left clicks \*/  if (button != GLUT\_LEFT\_BUTTON || state != GLUT\_DOWN)  return;  /\* 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;  /\* See if we have room for any more control points \*/  if (ncpts == MAX\_CPTS)  return;  /\* Save the point \*/  cpts[ncpts][0] = wx;  cpts[ncpts][1] = wy;  cpts[ncpts][2] = 0.0;  ncpts++;  /\* Draw the point \*/  glColor3f(0.0, 0.0, 0.0);  glPointSize(5.0);  glBegin(GL\_POINTS);  glVertex3f(wx, wy, 0.0);  glEnd();    glFlush();  }  /\* This routine handles keystroke commands \*/  void keyboard(unsigned char key, int x, int y)  {  static curveType lasttype = BEZIER;  switch (key)  {  case 'q': case 'Q':  exit(0);  break;  case 'c': case 'C':  ncpts = 0;  glutPostRedisplay();  break;  case 'e': case 'E':  glutPostRedisplay();  break;  case 'b': case 'B':  drawCurves(BEZIER);  lasttype = BEZIER;  break;  case 'i': case 'I':  drawCurves(INTERPOLATED);  lasttype = INTERPOLATED;  break;  case 's': case 'S':  drawCurves(BSPLINE);  lasttype = BSPLINE;  break;  }  }  /\* This routine handles window resizes \*/  void reshape(int w, int h)  {  width = w;  height = h;  /\* Set the transformations \*/  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);  }  main(int argc, char \*\*argv)  {  /\* Intialize the program \*/  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_RGB);  glutInitWindowSize(width, height);  glutCreateWindow("curves");  /\* Register the callbacks \*/  glutDisplayFunc(display);  glutMouseFunc(mouse);  glutKeyboardFunc(keyboard);  glutReshapeFunc(reshape);  glClearColor(1.0, 1.0, 1.0, 1.0);  glEnable(GL\_MAP1\_VERTEX\_3);  glutMainLoop();  } |

十、應用說明：

1. 應用程式目的:透過實際操作插入點與選擇要用 (1)Bezier curves(2)interpolating curves (3)B-spline curves 三種不同的方式來劃出曲線，讓使用者更加了解三者的差異，及曲線與控制點的關係。
2. DrawCurves(): 用控制點的資料來繪製曲線。
   1. ComputeMatrix(): 計算由控制點轉換的矩陣，會隨著使用方式的不同，用不同的計算方法。
3. typedef enum{ BEZIER,INTERPOLATED,BSPLINE} curveType:先定義三種曲線
4. #define MAX\_CPTS 25:定義控制點最多只能有25個

十一、注意事項：

1. 當控制點的數量小於4時，不能產生任何曲線，且當已經畫了曲線後又新增控制點時，會清除所有曲線並須重新繪製曲線。

十二、參考資料：

1. INTERACTIVE COMPUTER GRAPHICS A TOP-DOWN APPROACH WITH SHADER-BASED OPENGL 6th

https://inspirit.net.in/books/academic/Interactive%20Computer%20Graphics.pdf