**計算機圖學單元介紹**

一、英文主題：

Chapter 5: Viewing

二、中文主題：

單元05： 視角切換

三、組別：

第05組

四、組員：

B0729048 陳立翔；B0729067 陳希賢；B0829022 翁旻醇；B0829027 鄭星逸；

B0829048 李孟嘉；B0829056 陳杰；B0829060 黃志祥

五、作業分工：

(詳見作業報告) …略

六、功能簡述：

可以對相機的位置進行調整以獲得不同的畫面，可以從幾何元件的不同角度去觀察，得到不同角度的視角。

七、主要程式碼：

本單元為第一次程式實作，為完整解釋程式流程，故將全部原始碼詳列如下，

相關檔案：Ch\_05\_tm5\_src1.cpp

|  |
| --- |
| /\* Rotating cube with viewer movement from Chapter 5 \*/  /\* Cube definition and display similar to rotating--cube program \*/    /\* We use the Lookat function in the display callback to point  the viewer, whose position can be altered by the x,X,y,Y,z, and Z keys.  The perspective view is set in the reshape callback \*/    #include<stdlib.h>  #include<stdio.h>  #include<time.h>  #include <GL/glut.h>    GLfloat vertices[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},  {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},  {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};    GLfloat normals[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},  {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},  {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};    GLfloat colors[][3] = {{0.0,0.0,0.0},{1.0,0.0,0.0},  {1.0,1.0,0.0}, {0.0,1.0,0.0}, {0.0,0.0,1.0},  {1.0,0.0,1.0}, {1.0,1.0,1.0}, {0.0,1.0,1.0}};    void polygon(int a, int b, int c , int d)  {  glBegin(GL\_POLYGON);  glColor3fv(colors[a]);  glNormal3fv(normals[a]);  glVertex3fv(vertices[a]);  glColor3fv(colors[b]);  glNormal3fv(normals[b]);  glVertex3fv(vertices[b]);  glColor3fv(colors[c]);  glNormal3fv(normals[c]);  glVertex3fv(vertices[c]);  glColor3fv(colors[d]);  glNormal3fv(normals[d]);  glVertex3fv(vertices[d]);  glEnd();  }    void colorcube()  {  polygon(0,3,2,1);  polygon(2,3,7,6);  polygon(0,4,7,3);  polygon(1,2,6,5);  polygon(4,5,6,7);  polygon(0,1,5,4);  }    static GLfloat theta[] = {0.0,0.0,0.0};  static GLint axis = 2;  static GLdouble viewer[]= {0.0, 0.0, 5.0}; /\* initial viewer location \*/    void display(void)  {    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);    /\* Update viewer position in modelview matrix \*/    glLoadIdentity();  gluLookAt(viewer[0],viewer[1],viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);    /\* rotate cube \*/    glRotatef(theta[0], 1.0, 0.0, 0.0);  glRotatef(theta[1], 0.0, 1.0, 0.0);  glRotatef(theta[2], 0.0, 0.0, 1.0);    colorcube();    glFlush();  glutSwapBuffers();  }    void mouse(int btn, int state, int x, int y)  {  if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) axis = 0;  if(btn==GLUT\_MIDDLE\_BUTTON && state == GLUT\_DOWN) axis = 1;  if(btn==GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN) axis = 2;  theta[axis] += 2.0;  if( theta[axis] > 360.0 ) theta[axis] -= 360.0;  display();  }    void keys(unsigned char key, int x, int y)  {    /\* Use x, X, y, Y, z, and Z keys to move viewer \*/    if(key == 'x') viewer[0]-= 1.0;  if(key == 'X') viewer[0]+= 1.0;  if(key == 'y') viewer[1]-= 1.0;  if(key == 'Y') viewer[1]+= 1.0;  if(key == 'z') viewer[2]-= 1.0;  if(key == 'Z') viewer[2]+= 1.0;  display();  }    void myReshape(int w, int h)  {  glViewport(0, 0, w, h);    /\* Use a perspective view \*/    glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  if(w<=h) glFrustum(-2.0, 2.0, -2.0 \* (GLfloat) h/ (GLfloat) w,  2.0\* (GLfloat) h / (GLfloat) w, 2.0, 20.0);  else glFrustum(-2.0, 2.0, -2.0 \* (GLfloat) w/ (GLfloat) h,  2.0\* (GLfloat) w / (GLfloat) h, 2.0, 20.0);    /\* Or we can use gluPerspective \*/    /\* gluPerspective(45.0, w/h, -10.0, 10.0); \*/    glMatrixMode(GL\_MODELVIEW);  }    Int main(int argc, char \*\*argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(500, 500);  glutCreateWindow("colorcube");  glutReshapeFunc(myReshape);  glutDisplayFunc(display);  glutMouseFunc(mouse);  glutKeyboardFunc(keys);  glEnable(GL\_DEPTH\_TEST);  glutMainLoop();  }  執行結果： |

八、程式說明：

此程序執行時會打開一個窗口，將背景設為黑色，並繪製一個3D的彩色正方形於畫面中心。通過按鼠標左鍵、郵件、滾輪可以對立方體進行不同方向的旋轉。按鍵盤的大小寫x、y、z可以調整攝像機的x、y、z軸坐標，從不同的角度進行觀察。

(1)開頭引用 <GL/freeglut.h>函式庫，可以執行OpenGL的相關函數，其他函式庫用於數學計算與其他用途。

(2) glutInit(&argc, argv);這個函數用於初始化GLUT庫，對應main函數的形式是 glutInit(&argc, argv);代入進行初始化程式，如需使用其他目的可以在網路上查詢。

(3) glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA); 用來設定GLUT核心區域中的兩項狀態參數。GLUT\_DOUBLE和GLUT\_RGBA都是在<GL/freeglut.h>已定義的公用常數，但在執行時才設定給GLUT知道。GLUT\_DOUBLE 是指設置雙緩衝而GLUT\_RGBA 是指色彩模式是RGBA，多了一個透明度(A)，做此設定之後，如果用到色彩，就要用四維描述，比GLUT\_RGB多了一維。

(4) glutInitWindowSize(500, 500);設定視窗大小(w,h)這裡設定為500\*500的窗口。

(5) gluPerspective();gluPerspective 函式會將視圖分割指定為全局座標系統。 一般而言，gluPerspective 中的長寬比應該符合相 關聯之區的外觀比例。 例如， 外觀 = 2.0 表示檢視器的角度為 x 中的寬度兩倍，如同 y 一樣。 如果圖區的寬度為 高度，則會顯示影像，而不會失真。 GluPerspective 所產生的矩陣會乘以目前的矩陣，就像是使用產生的矩陣來呼叫 glMultMatrix一樣。 若要改 為將透視圖矩陣載入至目前的矩陣堆疊，請在呼叫 gluPerspective 之前呼叫 glLoadIdentity。

(6) void WINAPI glFrustum( GLdouble left, GLdouble right, Gldouble bottom,GLdoubletop, GLdouble zNear, GLdouble zFar ); left 左方垂直裁剪平面的座標。 right 右垂直裁剪平面的座標。 bottom 下水準裁剪平面的座標。 top 下水準裁剪平面的座標。 zNear 接近深度裁剪平面的距離。 必須是正數。 zFar 距離最深度裁剪平面的距離。 必須是正數。

(7) void WINAPI glOrtho( GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar ); ​left 左方垂直裁剪平面的座標。 right Theright 垂直裁剪平面的座標。bottom 底部水準裁剪平面的座標。 top 上方水準裁剪計畫的座標。 zNear 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。 zFar 距離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。​

(8) glutMouseFunc(mouse); 這函式的目的是用來設定滑鼠的回撥函式，此程序中通過按左鍵右鍵滾輪來進行不同軸的旋轉。

(9) glutKeyboardFunc(keys); 這函式的目的是用來設定鍵盤的回撥函式，次程序中可以按x, X, y, Y, z, Z來移動攝像機的位置。

(10) gluLookAt();gluLookAt 函式會建立衍生自眼睛點的視圖矩陣、指出場景中心的參考點，以及向上向量。 矩陣會將參考點對 應至負 Z 軸，並將眼睛點對應至原點，如此當您使用一般投射矩陣時，場景的中心就會對應到此區的中心。 同樣 地，上傳至「觀看」平面的向上向量所描述的方向，會對應至正 y 軸，使其在視口中向上指向。 向上向量不能平行 至從眼睛到參考點的可見線。 GluLookAt 所產生的矩陣會 postmultiplies 目前的矩陣。​

(11)glutDisplayFunc(display);這個API是把一段我們自已撰寫的程式 'display' 其程式碼的指標值(index)寫到GLUT核心區域的「待執行程序」位置。函數內傳函數指標，是一項高級的程式撰寫技巧，它在其他方面有各種變化與運用方式，而在OpenGL的常用技法中，我們藉此在GLUT的紀錄區域寫上我們所開發的程式碼，以待後續呼叫執行。

(12)glutMainLoop();

glutMainLoop() 是一個GLUT的動作API，它將控制權移送給GLUT，並開始它自己的內部循環。此前，我們曾用glutDisplayFunc(display)在GLUT的核心區域設定一段我們自己撰寫的程式 'display()' 作為它的「待執行程序」，現在GLUT取得控制權之後就會去執行這段程式。

glutMainLoop() 現在做兩件事：1.實際顯示出glutCreateWindow()所建立的視窗，2.執行之前指定的自撰程式 'display()'

九、延伸應用程式碼： Ch\_05\_tm5\_src2.cpp

|  |
| --- |
| /\* Rotating cube with viewer movement from Chapter 5 \*/  /\* Cube definition and display similar to rotating--cube program \*/    /\* We use the Lookat function in the display callback to point  the viewer, whose position can be altered by the x,X,y,Y,z, and Z keys.  The perspective view is set in the reshape callback \*/    #include<stdlib.h>  #include<stdio.h>  #include<time.h>  #include <GL/glut.h>  #include <math.h>  #include <cmath>      GLfloat vertices[][3] = {{0.0,0.0,0.0},{2.0,0.0,0.0},  {2.0,2.0,0.0}, {0.0,2.0,0.0}, {0.0,0.0,2.0},  {2.0,0.0,2.0}, {2.0,2.0,2.0}, {0.0,2.0,2.0}};    GLfloat normals[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},  {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},  {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};    GLfloat colors[][3] = {{0.0,0.0,0.0},{1.0,0.0,0.0},  {1.0,1.0,0.0}, {0.0,1.0,0.0}, {0.0,0.0,1.0},  {1.0,0.0,1.0}, {1.0,1.0,1.0}, {0.0,1.0,1.0}};  GLfloat GroundVertics[][3]{{-1,0,3},{-1,0,-1},{3,0,-1},{3,0,3}};    void polygon(int a, int b, int c , int d)  {  glBegin(GL\_POLYGON);  glColor3fv(colors[a]);  glNormal3fv(normals[a]);  glVertex3fv(vertices[a]);  glColor3fv(colors[b]);  glNormal3fv(normals[b]);  glVertex3fv(vertices[b]);  glColor3fv(colors[c]);  glNormal3fv(normals[c]);  glVertex3fv(vertices[c]);  glColor3fv(colors[d]);  glNormal3fv(normals[d]);  glVertex3fv(vertices[d]);  glEnd();  }    // draw the ground  void ground(int a, int b, int c , int d)  {  glBegin(GL\_POLYGON);  glColor3fv(colors[a]);  glNormal3fv(normals[a]);  glVertex3fv(GroundVertics[a]);  glColor3fv(colors[b]);  glNormal3fv(normals[b]);  glVertex3fv(GroundVertics[b]);  glColor3fv(colors[c]);  glNormal3fv(normals[c]);  glVertex3fv(GroundVertics[c]);  glColor3fv(colors[d]);  glNormal3fv(normals[d]);  glVertex3fv(GroundVertics[d]);  glEnd();  }    void colorcube()  {  polygon(0,3,2,1);  polygon(2,3,7,6);  polygon(0,4,7,3);  polygon(1,2,6,5);  polygon(4,5,6,7);  polygon(0,1,5,4);  ground(0,1,2,3);  }    class Camera {  public:  double theta; //position of x and z  double y; //y  double dTheta; //angle change  double dy; //y change  public:  //init  Camera() : theta(0), y(3), dTheta(0.04), dy(0.2) {}    double getX() { return 5 \* cos(theta); }  double getY() { return y; }  double getZ() { return 5 \* sin(theta); }  void moveRight() { theta += dTheta; }  void moveLeft() { theta -= dTheta; }  void moveUp() { y += dy; }  void moveDown() { y -= dy; }  };    Camera camera;        void display(void)  {    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);    /\* Update viewer position in modelview matrix \*/    glLoadIdentity();  gluLookAt(camera.getX(), camera.getY(), camera.getZ(), 1.0, 1.0, 1.0, 0.0, 1.0, 0.0);    colorcube();    glFlush();  glutSwapBuffers();  }      void onKey(int key, int , int )  {    /\* Use x, X, y, Y, z, and Z keys to move viewer \*/    switch (key) {  case GLUT\_KEY\_LEFT: camera.moveLeft(); break;  case GLUT\_KEY\_RIGHT: camera.moveRight(); break;  case GLUT\_KEY\_UP: camera.moveUp(); break;  case GLUT\_KEY\_DOWN: camera.moveDown(); break;  }  display();  }    void myReshape(int w, int h)  {  glViewport(0, 0, w, h);    /\* Use a perspective view \*/    glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  if(w<=h) glFrustum(-2.0, 2.0, -2.0 \* (GLfloat) h/ (GLfloat) w,  2.0\* (GLfloat) h / (GLfloat) w, 2.0, 20.0);  else glFrustum(-2.0, 2.0, -2.0 \* (GLfloat) w/ (GLfloat) h,  2.0\* (GLfloat) w / (GLfloat) h, 2.0, 20.0);    /\* Or we can use gluPerspective \*/    /\* gluPerspective(45.0, w/h, -10.0, 10.0); \*/    glMatrixMode(GL\_MODELVIEW);  }    int main(int argc, char \*\*argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(500, 500);  glutCreateWindow("colorcube");  glutReshapeFunc(myReshape);  glutDisplayFunc(display);  glutSpecialFunc(onKey);    glEnable(GL\_DEPTH\_TEST);    glutMainLoop();  }  執行結果： |

十、應用說明：

主要變動有以下幾項：

1.新增了一個y坐標是0四邊形作為地面，將立方體放在這個地面上，中心點(1,1,1)，邊長是2，可以在移動攝像機時更加直觀地看到視角的變化，體會與旋轉物體的不同之處。

2.新增了 class Camera，裡面包含攝像機所需的各個參數，移動攝像機會繞著cube的中心點(1,1,1)進行旋轉， Class Camera中使用了theta來控制x和z的坐標，使camera距離(1,y,1)的距離都是固定的，旋轉時改變theta的大小，用sin和cos函數分別計算得到x和z的坐標，使攝像機旋轉的軌跡會是一個圓。改變了攝像機的焦點，使攝像機會將立方體的中心作為焦點，將立方體的中心放到畫面中間。

3.改變了移動的攝像機的方式，用鍵盤的上下左右鍵來移動攝像機，更加符合使用直覺，更好地操控攝像機來獲得不同角度的視角。

十一、其他範例程式：

Ch\_05\_tm5\_src3.c

|  |
| --- |
| /\*  \* shadow.c  \* This program draws a polygon and its shadow.  \*/      #include<stdlib.h>  #include<stdio.h>  #include<time.h>  #include <GL/glut.h>    GLfloat theta=0.0;    void display(void)    {  /\* set clear color to white and clear window \*/    GLfloat light[3]={0.0, 10.0, 0.0};  GLfloat m[16];    double sin(), cos();    int i;  for(i=0;i<16;i++) m[i]=0.0;    light[0]=10.0\*sin((6.28/180.0)\*theta);  light[2]=10.0\*cos((6.28/180.0)\*theta);  m[0]=m[5]=m[10]=1.0;  m[7]=-1.0/light[1];  glClearColor (1.0, 1.0, 1.0, 0.0);  glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);    /\* set drawing/fill color to red \*/    glColor3f(1.0, 0.0, 0.0);    /\* set up standard orthogonal view with clipping \*/  /\* box as cube of side 2 centered at origin \*/    glMatrixMode (GL\_PROJECTION);  glLoadIdentity ();  glOrtho(-2.0, 2.0, -2.0, 2.0, -5.0, 5.0);    glMatrixMode(GL\_MODELVIEW);  glLoadIdentity();  gluLookAt(1.0,1.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0);    /\* define unit square polygon \*/    glBegin(GL\_POLYGON);  glVertex3f(-0.5, 0.5, -0.5);  glVertex3f(-0.5, 0.5, 0.5);  glVertex3f(0.5, 0.5, 0.5);  glVertex3f(0.5, 0.5, -0.5);  glEnd();    glPushMatrix();  glTranslatef(light[0], light[1],light[2]);  glMultMatrixf(m);  glTranslatef(-light[0], -light[1],-light[2]);  glColor3f(0.0,0.0,0.0);  glBegin(GL\_POLYGON);  glVertex3f(-0.5, 0.5, -0.5);  glVertex3f(-0.5, 0.5, 0.5);  glVertex3f(0.5, 0.5, 0.5);  glVertex3f(0.5, 0.5, -0.5);  glEnd();  glPopMatrix();    /\* Swap buffers \*/    glutSwapBuffers();    }    void myidle()  {  theta+=2.0;  if(theta>360.0) theta-=360;  glutPostRedisplay();  }    int main(int argc, char\*\* argv)  {    /\* Initialize mode and open a window in upper left corner of screen \*/  /\* Window title is name of program (arg[0]) \*/    glutInit(&argc,argv);  glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB |GLUT\_DEPTH);  glutInitWindowSize(500,500);  glutInitWindowPosition(0,0);  glutCreateWindow("shadow");  glutDisplayFunc(display);  glEnable(GL\_DEPTH\_TEST);  glutIdleFunc(myidle);  glutMainLoop();    } |

图片包含 游戏机

描述已自动生成

十二、注意事項：

1.glMatrixMode(GL\_MODELVIEW);設定模型視圖矩陣

2.glLoadIdentity()將觀察矩陣載入單位矩陣,相機為於原點,朝向為負Z軸,上方向量為正Y軸.

3.gluLookAt() 相對于原點在移動相機

|  |  |
| --- | --- |
| 函式 | 簡介 |
| void gluLookAt ( |  |
| GLdouble eyex, | 相機位置的x,y,z坐標 |
| GLdouble eyey, |
| GLdouble eyez, |
| GLdouble centerx, | 相機朝向的x,y,z坐標 |
| GLdouble centery, |
| GLdouble centerz, |
| GLdouble upx, | 上方向量為正Y軸 |
| GLdouble upy, |
| GLdouble upz); |

十三、參考資料：

公開資料