**計算機圖學單元介紹**

一、英文主題：

Chapter 4: Geometric Transformations​

二、中文主題：

單元04：幾何轉換

三、組別：

第01組

四、組員：

B0729001\_王奕凱；B0729002\_江尊瑋；B0729049\_黃浩恩；B0729028\_鄭楷翰；

B0729026\_施育銓；B0729059\_何亞翰；B0727246\_劉瑞和

五、作業分工：

(詳見作業報告)

六、功能簡述：

可以應用於對像以重新定位或調整它們大小的轉換操作,幾何元件的移動與旋轉等變換

七、主要程式碼：

相關檔案：Ch\_04\_tm3\_src1.cpp(cube.c)

|  |
| --- |
| /\* Rotating cube with color interpolation \*/  /\* Demonstration of use of homogeneous coordinate  transformations and simple data structure for representing  cube from Chapter 4 \*/  /\*Both normals and colors are assigned to the vertices \*/  /\*Cube is centered at origin so (unnormalized) normals  are the same as the vertex values \*/  #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)  {  /\* draw a polygon via list of vertices \*/  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(void)  {  /\* map vertices to faces \*/  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;  void display(void)  {  /\* display callback, clear frame buffer and z buffer,  rotate cube and draw, swap buffers \*/  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  glLoadIdentity();  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 spinCube()  {  /\* Idle callback, spin cube 2 degrees about selected axis \*/  theta[axis] += 2.0;  if( theta[axis] > 360.0 ) theta[axis] -= 360.0;  /\* display(); \*/  glutPostRedisplay();  }  void mouse(int btn, int state, int x, int y)  {  /\* mouse callback, selects an axis about which to rotate \*/  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;  }  void myReshape(int w, int h)  {  glViewport(0, 0, w, h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  if (w <= h)  glOrtho(-2.0, 2.0, -2.0 \* (GLfloat) h / (GLfloat) w,  2.0 \* (GLfloat) h / (GLfloat) w, -10.0, 10.0);  else  glOrtho(-2.0 \* (GLfloat) w / (GLfloat) h,  2.0 \* (GLfloat) w / (GLfloat) h, -2.0, 2.0, -10.0, 10.0);  glMatrixMode(GL\_MODELVIEW);  }  void  main(int argc, char \*\*argv)  {  glutInit(&argc, argv);  /\* need both double buffering and z buffer \*/  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(500, 500);  glutCreateWindow("colorcube");  glutReshapeFunc(myReshape);  glutDisplayFunc(display);  glutIdleFunc(spinCube);  glutMouseFunc(mouse);  glEnable(GL\_DEPTH\_TEST); /\* Enable hidden--surface--removal \*/  glutMainLoop();  } |

八、程式說明：

(1) 執行OpenGL時，在標頭檔部份，需引用<GL/freeglut.h>才能執行。

(2) 直接由 main() 主程式開始跑流程；

(3) glutInit(&argc, argv); 用來初始化GLUT，這裡它的參數直接由main()抄過來，main()執行時如果下達 '-sync' 或 '-gldebug' 等參數，就可以直接帶入進行初始化，進而達到監視及除錯等功能。

(4) glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

這個函數是用來設定GLUT核心區域中的兩項狀態的參數。在執行時才設定給GLUT知道。GLUT\_DOUBLE 是指設置雙緩衝而GLUT\_RGB 是指色彩模式是RGB，GLUT\_DEPTH則是使用深度緩衝器

(5)視窗大小(w,h) 設定

glutInitWindowSize(500, 500);

(6) 以上4個狀態值設定完成後，執行 glutCreateWindow()，就會建立一個繪圖視窗，也可以說是向系統取得一塊記憶體區段(緩衝區)，並在GLUT核心區域記錄一個index，完成畫布的註記。

(7) glutReshapeFunc (reshape);這程式的目的是將因視窗改變的圖像調整回來，先利用glViewport(0, 0, w, h)來設定擷取的影像在視窗的哪一個部分顯示，再利用glMatrixMode(GL\_PROJECTION)將影像投影，並透過glLoadIdentity()將其轉為識別矩陣，再利用gluOrtho2D (-2.0, 2.0, -2.0, 2.0)重新擷取圖像，並將其變為新的識別矩陣。

(8) glutDisplayFunc(display);

這個API是把一段我們撰寫的程式 '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。

void display() :

display callback, clear frame buffer and z buffer, rotate cube and draw, swap buffers.

(9) glutIdleFunc(spinCube);

設置全域的回呼函數，當沒有視窗事件到達時，GLUT程式功能可以執行幕後處理任務或連續動畫。如果啟用，這個idle function會被不斷調用，直到有視窗事件發生。

(10) glutMouseFunc (mouse);

這函式的目的是用來設定滑鼠的。

選擇要旋轉的軸(x,y,z)

(11) void colorcube();

透過呼叫polygon來畫出一個正方體。

void polygon(int a, int b, int c , int d); :

根據輸入的四個頂點來畫出對應的四邊形

(12) glutMainLoop();

glutMainLoop() 是一個GLUT的動作API，它將控制權移傳給GLUT，並開始自己的內部循環。前面曾用過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() 做兩件事：

1.顯示出glutCreateWindow()所建立的視窗

2.執行之前的自撰程式 'display()'

九、延伸應用程式碼： (Ch\_04\_tm3\_src2.cpp)(cubev.c)

|  |
| --- |
| #include<stdlib.h>  #include<stdio.h>  #include<time.h>  #include <GL/glut.h>  GLfloat vertices[] = {-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[] = {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};  GLubyte cubeIndices[]={0,3,2,1,2,3,7,6,0,4,7,3,1,2,6,5,4,5,6,7,0,1,5,4};  static GLfloat theta[] = {0.0,0.0,0.0};  static GLint axis = 2;  void display(void)  {  /\* display callback, clear frame buffer and z buffer,  rotate cube and draw, swap buffers \*/  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  glLoadIdentity();  gluLookAt(1.0,1.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0);  glTranslatef(0.0, 3.0, 0.0);  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);  glColorPointer(3,GL\_FLOAT, 0, colors);  glDrawElements(GL\_QUADS, 24, GL\_UNSIGNED\_BYTE, cubeIndices);  glutSwapBuffers();  }  void spinCube()  {  /\* Idle callback, spin cube 2 degrees about selected axis \*/  theta[axis] += 2.0;  if( theta[axis] > 360.0 ) theta[axis] -= 360.0;  glutPostRedisplay();  }  void mouse(int btn, int state, int x, int y)  {  /\* mouse callback, selects an axis about which to rotate \*/  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;  }  void myReshape(int w, int h)  {  glViewport(0, 0, w, h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  if (w <= h)  glOrtho(-4.0, 4.0, -3.0 \* (GLfloat) h / (GLfloat) w,  5.0 \* (GLfloat) h / (GLfloat) w, -10.0, 10.0);  else  glOrtho(-4.0 \* (GLfloat) w / (GLfloat) h,  4.0 \* (GLfloat) w / (GLfloat) h, -3.0, 5.0, -10.0, 10.0);  glMatrixMode(GL\_MODELVIEW);  }  void  main(int argc, char \*\*argv)  {  /\* need both double buffering and z buffer \*/  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(500, 500);  glutCreateWindow("colorcube");  glutReshapeFunc(myReshape);  glutDisplayFunc(display);  glutIdleFunc(spinCube);  glutMouseFunc(mouse);  glEnable(GL\_DEPTH\_TEST); /\* Enable hidden--surface--removal \*/  glEnableClientState(GL\_COLOR\_ARRAY);  glEnableClientState(GL\_VERTEX\_ARRAY);  glVertexPointer(3, GL\_FLOAT, 0, vertices);  glColorPointer(3,GL\_FLOAT, 0, colors);  glClearColor(1.0,1.0,1.0,1.0);  glColor3f(1.0,1.0,1.0);  glutMainLoop();  } |

十、應用說明：

基本上與cube.c一樣，不同之處在於cubev.c沒有使用polygon與colorcube這兩格函式來畫出正方體，而是利用頂點陣列來達成旋轉的功能。

十一、其他範例程式：

(無)。

十二、注意事項：

(無)。

十三、參考資料：

https://learnopengl.com/Getting-started/Transformations