**計算機圖學單元介紹**

1. 英文主題：

Chapter 6: Shading

1. 中文主題：

單元06：打光

1. 組別：

第05組

1. 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

1. 作業分工：

(詳見作業報告) …

1. 功能簡述：

本單元內容為製造不同光照效果

1. 主要程式碼：

相關檔案：Ch\_06\_tm5\_src1.cpp

|  |
| --- |
| /\* Recursive subdivision of cube (Chapter 6). Three display  modes: wire frame, constant, and interpolative shading \*/  /\*Program also illustrates defining materials and light sources  in myiit() \*/  /\* mode 0 = wire frame, mode 1 = constant shading,mode 1折線圖, mode 2 Phong著色法, mode 3強度插值明暗處理  mode 3 = interpolative shading \*/  //強度插值明暗處理最多細節 但繪畫速度也最慢 目前大部分的3A遊戲大作都是使用此方法  //折線圖最沒有細節但速度最快 適合運用在草圖  #include<stdlib.h>  #include<stdio.h>  #include<time.h>  #include <GL/glut.h>  typedef float point[4];  /\* initial tetrahedron \*/  point v[]={{0.0, 0.0, 1.0}, {0.0, 0.942809, -0.33333},  {-0.816497, -0.471405, -0.333333}, {0.816497, -0.471405, -0.333333}};  int n;  int mode;  void triangle( point a, point b, point c)  /\* display one triangle using a line loop for wire frame, a single  normal for constant shading, or three normals for interpolative shading \*/  {  if (mode==0) glBegin(GL\_LINE\_LOOP);//所有點連起來  else glBegin(GL\_POLYGON); //繪製單個多邊形。 頂點 1 到 N 定義了這個多邊形  if(mode==1) glNormal3fv(a); //a的內容當法線座標    if(mode==2) glNormal3fv(a); //a的內容當法線座標  glVertex3fv(a); //a的內容當頂點  if(mode==2) glNormal3fv(b); //b的內容當法線座標  glVertex3fv(b); //b的內容當頂點  if(mode==2) glNormal3fv(c); //c的內容當法線座標  glVertex3fv(c); //c的內容當頂點  glEnd();  }  void normal(point p)  {  /\* normalize a vector \*///邊長為1的多邊形  double sqrt();  float d =0.0;  int i;  for(i=0; i<3; i++) d+=p[i]\*p[i];  d=sqrt(d);//d=頂點座標平方相加再開根號 ，(0,0,0)到點(x,y,z)的距離  if(d>0.0) for(i=0; i<3; i++) p[i]=p[i]/d;//座標變成在1內  }  void divide\_triangle(point a, point b, point c, int m)  {  /\* triangle subdivision using vertex numbers  righthand rule applied to create outward pointing faces  使用頂點數進行三角形細分  右手定則應用於創建向外指向的面 \*/  point v1, v2, v3;  int j;  if(m>0)  {  for(j=0; j<3; j++) v1[j]=a[j]+b[j];  normal(v1);  for(j=0; j<3; j++) v2[j]=a[j]+c[j];  normal(v2);  for(j=0; j<3; j++) v3[j]=b[j]+c[j];  normal(v3);  divide\_triangle(a, v1, v2, m-1);  divide\_triangle(c, v2, v3, m-1);  divide\_triangle(b, v3, v1, m-1);  divide\_triangle(v1, v3, v2, m-1);  }  else(triangle(a,b,c)); /\* draw triangle at end of recursion \*/  }  void tetrahedron( int m)  {  /\* Apply triangle subdivision to faces of tetrahedron \*/    divide\_triangle(v[0], v[1], v[2], m);  divide\_triangle(v[3], v[2], v[1], m);  divide\_triangle(v[0], v[3], v[1], m);  divide\_triangle(v[0], v[2], v[3], m);  }  void display(void)  {  /\* Displays all three modes, side by side \*///輸出三種模式的圖  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  glLoadIdentity();  mode=0;  tetrahedron(n);  mode=1;  glTranslatef(-2.0, 0.0,0.0);  tetrahedron(n);  mode=2;  glTranslatef( 0.0, 0.0,0.0);  tetrahedron(n);  glFlush();  }  void myReshape(int w, int h)  {  glViewport(0, 0, w, h); //設定視窗高/寬度及位置  glMatrixMode(GL\_PROJECTION); //將後續矩陣運算應用於投影矩陣堆疊  glLoadIdentity(); //1個單位矩陣  if (w <= h) //glOrtho 函數將當前矩陣乘以正交矩陣。  glOrtho(-4.0, 4.0, -4.0 \* (GLfloat) h / (GLfloat) w,  4.0 \* (GLfloat) h / (GLfloat) w, -10.0, 10.0);  else  glOrtho(-4.0 \* (GLfloat) w / (GLfloat) h,  4.0 \* (GLfloat) w / (GLfloat) h, -4.0, 4.0, -10.0, 10.0);  glMatrixMode(GL\_MODELVIEW); //將後續矩陣操作應用於模型視圖矩陣堆疊  display();  }  void myinit()  {  GLfloat mat\_specular[]={1.0, 1.0, 1.0, 1.0};  GLfloat mat\_diffuse[]={1.0, 1.0, 1.0, 1.0};  GLfloat mat\_ambient[]={1.0, 1.0, 1.0, 1.0};  GLfloat mat\_shininess={100.0};  GLfloat light\_ambient[]={0.0, 0.0, 0.0, 1.0}; //環境反射光  GLfloat light\_diffuse[]={1.0, 1.0, 1.0, 1.0}; //漫反射光  GLfloat light\_specular[]={1.0, 1.0, 1.0, 1.0}; //鏡面反射光  /\* set up ambient, diffuse, and specular components for light 0 \*/  glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient); //燈的數量(0~7),選擇不同光源,第2參數要被設定的值  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);  glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);  /\* define material proerties for front face of all polygons \*/  glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular); //選前面後面，選面的材質，材質要被設定的值  glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);  glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);  glMaterialf(GL\_FRONT, GL\_SHININESS, mat\_shininess);  glShadeModel(GL\_SMOOTH); /\*enable smooth shading選擇平坦或平滑的著色。\*/  glEnable(GL\_LIGHTING); /\* enable lighting 啟用或禁用 OpenGL 功能\*/  glEnable(GL\_LIGHT0); /\* enable light 0 \*/  glEnable(GL\_DEPTH\_TEST); /\* enable z buffer \*/  glClearColor (1.0, 1.0, 1.0, 1.0);  glColor3f (0.0, 0.0, 0.0);  }  void  main(int argc, char \*\*argv)  {  n=5;  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(500, 500);  glutCreateWindow("sphere");  myinit();  glutReshapeFunc(myReshape);  glutDisplayFunc(display);  glutMainLoop();  } |

1. 程式說明：

此程序執行時會打開一個視窗，將背景設為黑色，用三種不同作法/光照，畫出三個球

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) glNormal3fv(a):設定頂點法線

(6) glVertex3fv(a)：設定頂點座標

(7) glOrtho():函數將當前矩陣乘以正交矩陣

(8) glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient); 燈的數量(0~7),選擇不同光源,第2參數要被設定的值

(9) glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular); //選前面後面，選面的材質，材質要被設定的值

(10) glShadeModel(GL\_SMOOTH);選擇平坦或平滑的著色

(11) glEnable(GL\_LIGHT0) : 開啟Light0照光

1. 延伸應用程式碼：

相關檔案：Ch\_06\_tm5\_src2.cpp

|  |
| --- |
| #include <GL/glut.h>  #include <stdlib.h>  void init(void)  {  //材質反光測試  GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 }; //鏡面反射參數  GLfloat mat\_shininess[] = { 50.0 }; //高光指數  GLfloat light\_position[] = { 1.0, 1.0, 1.0, 0.0 };  GLfloat white\_light[] = { 1.0, 1.0, 1.0, 1.0 }; //光的顏色  GLfloat Light\_Model\_Ambient[] = { 0.2, 0.2, 0.2, 1.0 }; //環境光參數  glClearColor(0.0, 0.0, 0.0, 0.0); //背景色  glShadeModel(GL\_SMOOTH); //FLAT or SMOOTH  //材質屬性  glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);  glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);  //燈光設置  glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, white\_light); //散射光屬性  glLightfv(GL\_LIGHT0, GL\_SPECULAR, white\_light); //鏡面反射光  glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, Light\_Model\_Ambient); //環境光參數  glEnable(GL\_LIGHTING); //開關:使用光  glEnable(GL\_LIGHT0); //打開0號燈  glEnable(GL\_DEPTH\_TEST); //打開深度  }  void display(void)  {  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  glutSolidTeapot(0.5);  glFlush(); //glSwapBuffers();  }  void reshape(int w, int h)  {  glViewport(0, 0, (GLsizei)w, (GLsizei)h);  //設置投影參數  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  //正交投影  if (w <= h)  glOrtho(-1.5, 1.5, -1.5\*(GLfloat)h / (GLfloat)w, 1.5\*(GLfloat)h / (GLfloat)w, -10.0, 10.0);  else  glOrtho(-1.5\*(GLfloat)w / (GLfloat)h, 1.5\*(GLfloat)w / (GLfloat)h, -1.5, 1.5, -10.0, 10.0);  //設置模型參數  glMatrixMode(GL\_MODELVIEW);  glLoadIdentity();  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(500, 500);  glutInitWindowPosition(100, 100);  glutCreateWindow("茶壺");  init();  glutDisplayFunc(display);  glutReshapeFunc(reshape);  glutMainLoop();  return 0;  } |

1. 應用說明：

畫一個茶壺來模擬光照

十一、注意事項：此為src1的延伸，注意使用src1相關函數就不會有問題

十二、參考資料：

網路上公開資料