**計算機圖學單元介紹**

1. 英文主題：

Chapter 10: Modeling and Procedural Methods

1. 中文主題：

單元10：模組化與程序化

1. 組別：

第8組

1. 組員：

B0829013\_林冠斌；B0829017\_張睿；B0829029\_劉建良；B0829035\_莊淙賢；

B0829054\_李彥廷；B0829055\_毛聖淇；

1. 功能簡述：

例： 本單元內容為介紹模組搭建，包含利用資料結構和物理、數學特性等方式，製作出現實世界存在之物品的骨架。

1. 主要程式碼：

相關檔案：Ch\_10\_tm8\_src1.cpp

|  |
| --- |
| (以1x1表格填寫，文字為 “Segoe UI” 11點字，固定行高12點，內容可變更文字顏色)  /\* Interactive Figure Program from Chapter 8 using cylinders (quadrics) \*/  /\* Style similar to robot program but here we must traverse tree to display \*/  /\* Cylinders are displayed as filled and light/material properties \*/  /\* are set as in sphere approximation program \*/  #include<stdlib.h>  #include <GL/glut.h>  #define TORSO\_HEIGHT 5.0  #define UPPER\_ARM\_HEIGHT 3.0  #define LOWER\_ARM\_HEIGHT 2.0  #define UPPER\_LEG\_RADIUS 0.5  #define LOWER\_LEG\_RADIUS 0.5  #define LOWER\_LEG\_HEIGHT 2.0  #define UPPER\_LEG\_HEIGHT 3.0  #define UPPER\_LEG\_RADIUS 0.5  #define TORSO\_RADIUS 1.0  #define UPPER\_ARM\_RADIUS 0.5  #define LOWER\_ARM\_RADIUS 0.5  #define HEAD\_HEIGHT 1.5  #define HEAD\_RADIUS 1.0  void head();  void torso();  void left\_upper\_arm();  void right\_upper\_arm();  void left\_upper\_leg();  void right\_upper\_leg();  typedef float point[3];  typedef struct treenode  {  GLfloat m[16];  void (\*f)();  struct treenode \*sibling;  struct treenode \*child;  }treenode, \*t\_ptr;  static GLfloat theta[11] = {0.0,0.0,0.0,0.0,0.0,0.0,0.0,  180.0,0.0,180.0,0.0}; /\* initial joint angles \*/  static GLint angle = 2;  GLUquadricObj \*t, \*h, \*lua, \*lla, \*rua, \*rla, \*lll, \*rll, \*rul, \*lul;  double size=1.0;  t\_ptr torso\_ptr, head\_ptr, lua\_ptr, rua\_ptr, lll\_ptr, rll\_ptr,  lla\_ptr, rla\_ptr, rul\_ptr, lul\_ptr;  void traverse(t\_ptr root)  {  if(root==NULL) return;  glPushMatrix();  glMultMatrixf(root->m);  root->f();  if(root->child!=NULL) traverse(root->child);  glPopMatrix();  if(root->sibling!=NULL) traverse(root->sibling);  }  void torso()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(t,TORSO\_RADIUS, TORSO\_RADIUS, TORSO\_HEIGHT,10,10);  glPopMatrix();  }  void head()  {  glPushMatrix();  glTranslatef(0.0, 0.5\*HEAD\_HEIGHT,0.0);  glScalef(HEAD\_RADIUS, HEAD\_HEIGHT, HEAD\_RADIUS);  gluSphere(h,1.0,10,10);  glPopMatrix();  }  void left\_upper\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lua,UPPER\_ARM\_RADIUS, UPPER\_ARM\_RADIUS, UPPER\_ARM\_HEIGHT,10,10);  glPopMatrix();  }  void left\_lower\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lla,LOWER\_ARM\_RADIUS, LOWER\_ARM\_RADIUS, LOWER\_ARM\_HEIGHT,10,10);  glPopMatrix();  }  void right\_upper\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rua,UPPER\_ARM\_RADIUS, UPPER\_ARM\_RADIUS, UPPER\_ARM\_HEIGHT,10,10);  glPopMatrix();  }  void right\_lower\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rla,LOWER\_ARM\_RADIUS, LOWER\_ARM\_RADIUS, LOWER\_ARM\_HEIGHT,10,10);  glPopMatrix();  }  void left\_upper\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lul,UPPER\_LEG\_RADIUS, UPPER\_LEG\_RADIUS, UPPER\_LEG\_HEIGHT,10,10);  glPopMatrix();  }  void left\_lower\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lll,LOWER\_LEG\_RADIUS, LOWER\_LEG\_RADIUS, LOWER\_LEG\_HEIGHT,10,10);  glPopMatrix();  }  void right\_upper\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rul,UPPER\_LEG\_RADIUS, UPPER\_LEG\_RADIUS, UPPER\_LEG\_HEIGHT,10,10);  glPopMatrix();  }  void right\_lower\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rll,LOWER\_LEG\_RADIUS, LOWER\_LEG\_RADIUS, LOWER\_LEG\_HEIGHT,10,10);  glPopMatrix();  }  void  display(void)  {  glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);  glLoadIdentity();  glColor3f(1.0, 0.0, 0.0);  traverse(torso\_ptr);  glutSwapBuffers();  }  void mouse(int btn, int state, int x, int y)  {  if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)  {  theta[angle] += 5.0;  if( theta[angle] > 360.0 ) theta[angle] -= 360.0;  }  if(btn==GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN)  {  theta[angle] -= 5.0;  if( theta[angle] < 360.0 ) theta[angle] += 360.0;  }  glPushMatrix();  switch(angle)  {  case 0 :  glLoadIdentity();  glRotatef(theta[0], 0.0, 1.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,torso\_ptr->m);  break;  case 1 : case 2 :  glLoadIdentity();  glTranslatef(0.0, TORSO\_HEIGHT+0.5\*HEAD\_HEIGHT, 0.0);  glRotatef(theta[1], 1.0, 0.0, 0.0);  glRotatef(theta[2], 0.0, 1.0, 0.0);  glTranslatef(0.0, -0.5\*HEAD\_HEIGHT, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,head\_ptr->m);  break;  case 3 :  glLoadIdentity();  glTranslatef(-(TORSO\_RADIUS+UPPER\_ARM\_RADIUS), 0.9\*TORSO\_HEIGHT, 0.0);  glRotatef(theta[3], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,lua\_ptr->m);  break;  case 5 :  glLoadIdentity();  glTranslatef(TORSO\_RADIUS+UPPER\_ARM\_RADIUS, 0.9\*TORSO\_HEIGHT, 0.0);  glRotatef(theta[5], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,rua\_ptr->m);  break;  case 9 :  glLoadIdentity();  glTranslatef(TORSO\_RADIUS+UPPER\_LEG\_RADIUS, 0.1\*UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[9], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,rul\_ptr->m);  break;  case 7 :  glLoadIdentity();  glTranslatef(-(TORSO\_RADIUS+UPPER\_LEG\_RADIUS), 0.1\*UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[7], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,lul\_ptr->m);  break;  case 4 :  glLoadIdentity();  glTranslatef(0.0, UPPER\_ARM\_HEIGHT, 0.0);  glRotatef(theta[4], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,lla\_ptr->m);  break;  case 8 :  glLoadIdentity();  glTranslatef(0.0, UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[8], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,lll\_ptr->m);  break;  case 10 :  glLoadIdentity();  glTranslatef(0.0, UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[10], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,rll\_ptr->m);  break;  case 6 :  glLoadIdentity();  glTranslatef(0.0, UPPER\_ARM\_HEIGHT, 0.0);  glRotatef(theta[6], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,rla\_ptr->m);  break;  }  glPopMatrix();  glutPostRedisplay();  }  void menu(int id)  {  if(id <11 ) angle=id;  if(id ==11 ) exit(0);  }  void  myReshape(int w, int h)  {  glViewport(0, 0, w, h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  if (w <= h)  glOrtho(-10.0, 10.0, -10.0 \* (GLfloat) h / (GLfloat) w,  10.0 \* (GLfloat) h / (GLfloat) w, -10.0, 10.0);  else  glOrtho(-10.0 \* (GLfloat) w / (GLfloat) h,  10.0 \* (GLfloat) w / (GLfloat) h, 0.0, 10.0, -10.0, 10.0);  glMatrixMode(GL\_MODELVIEW);  glLoadIdentity();  }  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};  GLfloat light\_diffuse[]={1.0, 0.0, 0.0, 1.0};  GLfloat light\_specular[]={1.0, 1.0, 1.0, 1.0};  GLfloat light\_position[]={10.0, 10.0, 10.0, 0.0};  glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);  glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);  glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);  glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);  glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);  glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);  glMaterialf(GL\_FRONT, GL\_SHININESS, mat\_shininess);  glShadeModel(GL\_SMOOTH);  glEnable(GL\_LIGHTING);  glEnable(GL\_LIGHT0);  glDepthFunc(GL\_LEQUAL);  glEnable(GL\_DEPTH\_TEST);  glClearColor(1.0, 1.0, 1.0, 1.0);  glColor3f(1.0, 0.0, 0.0);  /\* allocate quadrics with filled drawing style \*/  h=gluNewQuadric();  gluQuadricDrawStyle(h, GLU\_FILL);  t=gluNewQuadric();  gluQuadricDrawStyle(t, GLU\_FILL);  lua=gluNewQuadric();  gluQuadricDrawStyle(lua, GLU\_FILL);  lla=gluNewQuadric();  gluQuadricDrawStyle(lla, GLU\_FILL);  rua=gluNewQuadric();  gluQuadricDrawStyle(rua, GLU\_FILL);  rla=gluNewQuadric();  gluQuadricDrawStyle(rla, GLU\_FILL);  lul=gluNewQuadric();  gluQuadricDrawStyle(lul, GLU\_FILL);  lll=gluNewQuadric();  gluQuadricDrawStyle(lll, GLU\_FILL);  rul=gluNewQuadric();  gluQuadricDrawStyle(rul, GLU\_FILL);  rll=gluNewQuadric();  gluQuadricDrawStyle(rll, GLU\_FILL);  /\* Set up tree \*/  torso\_ptr = malloc(sizeof(treenode));  head\_ptr = malloc(sizeof(treenode));  lua\_ptr = malloc(sizeof(treenode));  rua\_ptr = malloc(sizeof(treenode));  lll\_ptr = malloc(sizeof(treenode));  rll\_ptr = malloc(sizeof(treenode));  lla\_ptr = malloc(sizeof(treenode));  rla\_ptr = malloc(sizeof(treenode));  rul\_ptr = malloc(sizeof(treenode));  lul\_ptr = malloc(sizeof(treenode));  glLoadIdentity();  glRotatef(theta[0], 0.0, 1.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,torso\_ptr->m);  torso\_ptr->f = torso;  torso\_ptr->sibling = NULL;  torso\_ptr->child = head\_ptr;  glLoadIdentity();  glTranslatef(0.0, TORSO\_HEIGHT+0.5\*HEAD\_HEIGHT, 0.0);  glRotatef(theta[1], 1.0, 0.0, 0.0);  glRotatef(theta[2], 0.0, 1.0, 0.0);  glTranslatef(0.0, -0.5\*HEAD\_HEIGHT, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,head\_ptr->m);  head\_ptr->f = head;  head\_ptr->sibling = lua\_ptr;  head\_ptr->child = NULL;  glLoadIdentity();  glTranslatef(-(TORSO\_RADIUS+UPPER\_ARM\_RADIUS), 0.9\*TORSO\_HEIGHT, 0.0);  glRotatef(theta[3], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,lua\_ptr->m);  lua\_ptr->f = left\_upper\_arm;  lua\_ptr->sibling = rua\_ptr;  lua\_ptr->child = lla\_ptr;  glLoadIdentity();  glTranslatef(TORSO\_RADIUS+UPPER\_ARM\_RADIUS, 0.9\*TORSO\_HEIGHT, 0.0);  glRotatef(theta[5], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,rua\_ptr->m);  rua\_ptr->f = right\_upper\_arm;  rua\_ptr->sibling = lul\_ptr;  rua\_ptr->child = rla\_ptr;  glLoadIdentity();  glTranslatef(-(TORSO\_RADIUS+UPPER\_LEG\_RADIUS), 0.1\*UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[7], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,lul\_ptr->m);  lul\_ptr->f = left\_upper\_leg;  lul\_ptr->sibling = rul\_ptr;  lul\_ptr->child = lll\_ptr;  glLoadIdentity();  glTranslatef(TORSO\_RADIUS+UPPER\_LEG\_RADIUS, 0.1\*UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[9], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,rul\_ptr->m);  rul\_ptr->f = right\_upper\_leg;  rul\_ptr->sibling = NULL;  rul\_ptr->child = rll\_ptr;  glLoadIdentity();  glTranslatef(0.0, UPPER\_ARM\_HEIGHT, 0.0);  glRotatef(theta[4], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,lla\_ptr->m);  lla\_ptr->f = left\_lower\_leg;  lla\_ptr->sibling = NULL;  lla\_ptr->child = NULL;  glLoadIdentity();  glTranslatef(0.0, UPPER\_ARM\_HEIGHT, 0.0);  glRotatef(theta[6], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,rla\_ptr->m);  rla\_ptr->f = right\_lower\_arm;  rla\_ptr->sibling = NULL;  rla\_ptr->child = NULL;  glLoadIdentity();  glTranslatef(0.0, UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[8], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,lll\_ptr->m);  lll\_ptr->f = left\_lower\_leg;  lll\_ptr->sibling = NULL;  lll\_ptr->child = NULL;  glLoadIdentity();  glTranslatef(0.0, UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[10], 1.0, 0.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,rll\_ptr->m);  rll\_ptr->f = right\_lower\_leg;  rll\_ptr->sibling = NULL;  rll\_ptr->child = NULL;  glLoadIdentity();    }  void main(int argc, char \*\*argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(500, 500);  glutCreateWindow("robot");  myinit();  glutReshapeFunc(myReshape);  glutDisplayFunc(display);  glutMouseFunc(mouse);  glutCreateMenu(menu);  glutAddMenuEntry("torso", 0);  glutAddMenuEntry("head1", 1);  glutAddMenuEntry("head2", 2);  glutAddMenuEntry("right\_upper\_arm", 3);  glutAddMenuEntry("right\_lower\_arm", 4);  glutAddMenuEntry("left\_upper\_arm", 5);  glutAddMenuEntry("left\_lower\_arm", 6);  glutAddMenuEntry("right\_upper\_leg", 7);  glutAddMenuEntry("right\_lower\_leg", 8);  glutAddMenuEntry("left\_upper\_leg", 9);  glutAddMenuEntry("left\_lower\_leg", 10);  glutAddMenuEntry("quit", 11);  glutAttachMenu(GLUT\_MIDDLE\_BUTTON);  glutMainLoop();  } |

1. 程式說明：

* Define

|  |  |
| --- | --- |
| #define TORSO\_HEIGHT 5.0 | 設定軀幹的長度 |
| #define UPPER\_ARM\_HEIGHT 3.0 | 設定上手臂長度 |
| #define LOWER\_ARM\_HEIGHT 2.0 | 設定下手臂長度 |
| #define UPPER\_LEG\_RADIUS 0.5 | 設定大腿半徑 |
| #define LOWER\_LEG\_RADIUS 0.5 | 設定小腿半徑 |
| #define LOWER\_LEG\_HEIGHT 2.0 | 設定小腿長度 |
| #define UPPER\_LEG\_HEIGHT 3.0 | 設定大腿長度 |
| #define TORSO\_RADIUS 1.0 | 設定軀幹半徑 |
| #define UPPER\_ARM\_RADIUS 0.5 | 設定上手臂半徑 |
| #define LOWER\_ARM\_RADIUS 0.5 | 設定下手臂半徑 |
| #define HEAD\_HEIGHT 1.5 | 設定頭部高度 |
| #define HEAD\_RADIUS 1.0 | 設定頭部半徑 |
| struct treenode | 存放要乘的陣列、兄弟和子孫 |
| GLfloat theta[11] | 存放改變的角度 |
| GLint angle | 存放按一次滑鼠會產生多少角度的改變 |

* 函式介紹

|  |  |
| --- | --- |
| void head()  {  glPushMatrix();  glTranslatef(0.0, 0.5\*HEAD\_HEIGHT,0.0);  glScalef(HEAD\_RADIUS, HEAD\_HEIGHT, HEAD\_RADIUS);  gluSphere(h,1.0,10,10);  glPopMatrix();  } | Translatef來產生matrix的平移，以此來製作頭部轉動;Scalef進行matrix的縮放;Sphere來製作頭部。 |
| void torso()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(t,TORSO\_RADIUS, TORSO\_RADIUS, TORSO\_HEIGHT,10,10);  glPopMatrix();  } | 用pushMatrix輸入目前位置資訊、Rotatef做角度移動、PopMatrix讓stack返回這個子節點進入之前的狀態、cylinder為圓柱的製作 |
| void left\_upper\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lua,UPPER\_ARM\_RADIUS, UPPER\_ARM\_RADIUS, UPPER\_ARM\_HEIGHT,10,10);  glPopMatrix();  } | 用pushMatrix輸入目前位置資訊、Rotatef做角度移動、PopMatrix讓stack返回這個子節點進入之前的狀態、cylinder為圓柱的製作，用來製作左上手臂 |
| void right\_upper\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rua,UPPER\_ARM\_RADIUS, UPPER\_ARM\_RADIUS, UPPER\_ARM\_HEIGHT,10,10);  glPopMatrix();  } | 用pushMatrix輸入目前位置資訊、Rotatef做角度移動、PopMatrix讓stack返回這個子節點進入之前的狀態、cylinder為圓柱的製作，用來製作右上手臂 |
| void left\_lower\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lla,LOWER\_ARM\_RADIUS, LOWER\_ARM\_RADIUS, LOWER\_ARM\_HEIGHT,10,10);  glPopMatrix();  } | 用pushMatrix輸入目前位置資訊、Rotatef做角度移動、PopMatrix讓stack返回這個子節點進入之前的狀態、cylinder為圓柱的製作，用來製作左下手臂 |
| void right\_lower\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rla,LOWER\_ARM\_RADIUS, LOWER\_ARM\_RADIUS, LOWER\_ARM\_HEIGHT,10,10);  glPopMatrix();  } | 用pushMatrix輸入目前位置資訊、Rotatef做角度移動、PopMatrix讓stack返回這個子節點進入之前的狀態、cylinder為圓柱的製作，用來製作右下手臂 |
| void left\_upper\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lul,UPPER\_LEG\_RADIUS, UPPER\_LEG\_RADIUS, UPPER\_LEG\_HEIGHT,10,10);  glPopMatrix();  } | 用pushMatrix輸入目前位置資訊、Rotatef做角度移動、PopMatrix讓stack返回這個子節點進入之前的狀態、cylinder為圓柱的製作，用來製作左大腿 |
| void left\_lower\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lll,LOWER\_LEG\_RADIUS, LOWER\_LEG\_RADIUS, LOWER\_LEG\_HEIGHT,10,10);  glPopMatrix();  } | 用pushMatrix輸入目前位置資訊、Rotatef做角度移動、PopMatrix讓stack返回這個子節點進入之前的狀態、cylinder為圓柱的製作，用來製作左小腿 |
| void right\_upper\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rul,UPPER\_LEG\_RADIUS, UPPER\_LEG\_RADIUS, UPPER\_LEG\_HEIGHT,10,10);  glPopMatrix();  } | 用pushMatrix輸入目前位置資訊、Rotatef做角度移動、PopMatrix讓stack返回這個子節點進入之前的狀態、cylinder為圓柱的製作，用來製作右大腿 |
| void right\_lower\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rll,LOWER\_LEG\_RADIUS, LOWER\_LEG\_RADIUS, LOWER\_LEG\_HEIGHT,10,10);  glPopMatrix();} | 用pushMatrix輸入目前位置資訊、Rotatef做角度移動、PopMatrix讓stack返回這個子節點進入之前的狀態、cylinder為圓柱的製作，用來製作右小腿 |
| void mouse(int btn, int state, int x, int y)  {  if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)  {  theta[angle] += 5.0;  if( theta[angle] > 360.0 ) theta[angle] -= 360.0;  }  if(btn==GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN)  {  theta[angle] -= 5.0;  if( theta[angle] < 360.0 ) theta[angle] += 360.0;  }  glPushMatrix();  switch(angle)  {  case 0 :  glLoadIdentity();  glRotatef(theta[0], 0.0, 1.0, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,torso\_ptr->m);  break;  case 1 : case 2 :  glLoadIdentity();  glTranslatef(0.0, TORSO\_HEIGHT+0.5\*HEAD\_HEIGHT, 0.0);  glRotatef(theta[1], 1.0, 0.0, 0.0);  glRotatef(theta[2], 0.0, 1.0, 0.0);  glTranslatef(0.0, -0.5\*HEAD\_HEIGHT, 0.0);  glGetFloatv(GL\_MODELVIEW\_MATRIX,head\_ptr->m);  break; | 利用case以及滑鼠偵測來確定我們想要改變部位，先glLoadIdentity()來抓整個物件的資料，glRotatef、glTranslatef來輸入要轉動的角度以及位置 |

1. 延伸應用程式碼：

相關檔案：Ch\_10\_tm8\_src2.cpp

|  |
| --- |
| (以1x1表格填寫，文字為 “Segoe UI” 11點字，固定行高12點，內容可變更文字顏色)  /\* Interactive Figure Program from Chapter 8 using cylinders (quadrics) \*/  /\* Style similar to robot program but here we must traverse tree to display \*/  /\* Cylinders are displayed as filled and light/material properties \*/  /\* are set as in sphere approximation program \*/  #include <stdlib.h>  #include <GL/glut.h>  #define TORSO\_HEIGHT 5.0  #define UPPER\_ARM\_HEIGHT 3.0  #define LOWER\_ARM\_HEIGHT 2.0  #define UPPER\_LEG\_RADIUS 0.5  #define LOWER\_LEG\_RADIUS 0.5  #define LOWER\_LEG\_HEIGHT 2.0  #define UPPER\_LEG\_HEIGHT 3.0  #define UPPER\_LEG\_RADIUS 0.5  #define TORSO\_RADIUS 1.0  #define UPPER\_ARM\_RADIUS 0.5  #define LOWER\_ARM\_RADIUS 0.5  #define HEAD\_HEIGHT 1.5  #define HEAD\_RADIUS 1.0  #define lower\_sword\_radius 1  #define lower\_sword\_height 0.5  #define upper\_sword\_radius 0.2  #define upper\_sword\_height 8  typedef float point[3];  static GLfloat theta[13] = {0.0,0.0,0.0,0.0,0.0,0.0,0.0,  180.0,0.0,180.0,0.0,0.0,0.0}; /\* initial joint angles \*/  static GLint angle = 2;  GLUquadricObj \*t, \*h, \*lua, \*lla, \*rua, \*rla, \*lll, \*rll, \*rul, \*lul, \*ls,\*us;  double size=1.0;  void torso()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(t,TORSO\_RADIUS, TORSO\_RADIUS, TORSO\_HEIGHT,10,10);  glPopMatrix();  }  void head()  {  glPushMatrix();  glTranslatef(0.0, 0.5\*HEAD\_HEIGHT,0.0);  glScalef(HEAD\_RADIUS, HEAD\_HEIGHT, HEAD\_RADIUS);  gluSphere(h,1.0,10,10);  glPopMatrix();  }  void left\_upper\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lua,UPPER\_ARM\_RADIUS, UPPER\_ARM\_RADIUS, UPPER\_ARM\_HEIGHT,10,10);  glPopMatrix();  }  void left\_lower\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lla,LOWER\_ARM\_RADIUS, LOWER\_ARM\_RADIUS, LOWER\_ARM\_HEIGHT,10,10);  glPopMatrix();  }  void right\_upper\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rua,UPPER\_ARM\_RADIUS, UPPER\_ARM\_RADIUS, UPPER\_ARM\_HEIGHT,10,10);  glPopMatrix();  }  void right\_lower\_arm()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rla,LOWER\_ARM\_RADIUS, LOWER\_ARM\_RADIUS, LOWER\_ARM\_HEIGHT,10,10);  glPopMatrix();  }  void left\_upper\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lul,UPPER\_LEG\_RADIUS, UPPER\_LEG\_RADIUS, UPPER\_LEG\_HEIGHT,10,10);  glPopMatrix();  }  void left\_lower\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(lll,LOWER\_LEG\_RADIUS, LOWER\_LEG\_RADIUS, LOWER\_LEG\_HEIGHT,10,10);  glPopMatrix();  }  void right\_upper\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rul,UPPER\_LEG\_RADIUS, UPPER\_LEG\_RADIUS, UPPER\_LEG\_HEIGHT,10,10);  glPopMatrix();  }  void right\_lower\_leg()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(rll,LOWER\_LEG\_RADIUS, LOWER\_LEG\_RADIUS, LOWER\_LEG\_HEIGHT,10,10);  glPopMatrix();  }  void lower\_sword()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(ls,lower\_sword\_radius, lower\_sword\_radius, lower\_sword\_height,10,10);  glPopMatrix();  }  void upper\_sword()  {  glPushMatrix();  glRotatef(-90.0, 1.0, 0.0, 0.0);  gluCylinder(ls,upper\_sword\_radius, upper\_sword\_radius, upper\_sword\_height,10,10);  glPopMatrix();  }  void  display(void)  {  glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);  glLoadIdentity();  glColor3f(1.0, 0.0, 0.0);  glRotatef(theta[0], 0.0, 1.0, 0.0);  torso();  glPushMatrix();  glTranslatef(0.0, TORSO\_HEIGHT+0.5\*HEAD\_HEIGHT, 0.0);  glRotatef(theta[1], 1.0, 0.0, 0.0);  glRotatef(theta[2], 0.0, 1.0, 0.0);  glTranslatef(0.0, -0.5\*HEAD\_HEIGHT, 0.0);  head();  glPopMatrix();  glPushMatrix();  glTranslatef(-(TORSO\_RADIUS+UPPER\_ARM\_RADIUS), 0.9\*TORSO\_HEIGHT, 0.0);  glRotatef(theta[3], 1.0, 0.0, 0.0);  left\_upper\_arm();  glTranslatef(0.0, UPPER\_ARM\_HEIGHT, 0.0);  glRotatef(theta[4], 1.0, 0.0, 0.0);  left\_lower\_arm();    glTranslatef(0.0, LOWER\_ARM\_HEIGHT, 0.0);  glRotatef(theta[11], 1.0, 0.0, 0.0);  lower\_sword();    glTranslatef(0.0, lower\_sword\_height, 0.0);  glRotatef(theta[12], 1.0, 0.0, 0.0);  upper\_sword();  glPopMatrix();  glPushMatrix();  glTranslatef(TORSO\_RADIUS+UPPER\_ARM\_RADIUS, 0.9\*TORSO\_HEIGHT, 0.0);  glRotatef(theta[5], 1.0, 0.0, 0.0);  right\_upper\_arm();  glTranslatef(0.0, UPPER\_ARM\_HEIGHT, 0.0);  glRotatef(theta[6], 1.0, 0.0, 0.0);  right\_lower\_arm();  glPopMatrix();  glPushMatrix();  glTranslatef(-(TORSO\_RADIUS+UPPER\_LEG\_RADIUS), 0.1\*UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[7], 1.0, 0.0, 0.0);  left\_upper\_leg();    glTranslatef(0.0, UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[8], 1.0, 0.0, 0.0);  left\_lower\_leg();  glPopMatrix();  glPushMatrix();  glTranslatef(TORSO\_RADIUS+UPPER\_LEG\_RADIUS, 0.1\*UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[9], 1.0, 0.0, 0.0);  right\_upper\_leg();  glTranslatef(0.0, UPPER\_LEG\_HEIGHT, 0.0);  glRotatef(theta[10], 1.0, 0.0, 0.0);  right\_lower\_leg();  glPopMatrix();  glFlush();  glutSwapBuffers();  }  void mouse(int btn, int state, int x, int y)  {  if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)  {  theta[angle] += 5.0;  if( theta[angle] > 360.0 ) theta[angle] -= 360.0;  }  if(btn==GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN)  {  theta[angle] -= 5.0;  if( theta[angle] < 360.0 ) theta[angle] += 360.0;  }  display();  }  void menu(int id)  {  if(id <11 ) angle=id;  if(id ==11 ) exit(0);  }  void  myReshape(int w, int h)  {  glViewport(0, 0, w, h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  if (w <= h)  glOrtho(-10.0, 10.0, -10.0 \* (GLfloat) h / (GLfloat) w,  10.0 \* (GLfloat) h / (GLfloat) w, -10.0, 10.0);  else  glOrtho(-10.0 \* (GLfloat) w / (GLfloat) h,  10.0 \* (GLfloat) w / (GLfloat) h, 0.0, 10.0, -10.0, 10.0);  glMatrixMode(GL\_MODELVIEW);  glLoadIdentity();  }  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};  GLfloat light\_diffuse[]={1.0, 0.0, 0.0, 1.0};  GLfloat light\_specular[]={1.0, 1.0, 1.0, 1.0};  GLfloat light\_position[]={10.0, 10.0, 10.0, 0.0};  glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);  glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);  glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);  glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);  glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);  glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);  glMaterialf(GL\_FRONT, GL\_SHININESS, mat\_shininess);  glShadeModel(GL\_SMOOTH);  glEnable(GL\_LIGHTING);  glEnable(GL\_LIGHT0);  glDepthFunc(GL\_LEQUAL);  glEnable(GL\_DEPTH\_TEST);  glClearColor(1.0, 1.0, 1.0, 1.0);  glColor3f(1.0, 0.0, 0.0);  /\* allocate quadrics with filled drawing style \*/  h=gluNewQuadric();  gluQuadricDrawStyle(h, GLU\_FILL);  t=gluNewQuadric();  gluQuadricDrawStyle(t, GLU\_FILL);  lua=gluNewQuadric();  gluQuadricDrawStyle(lua, GLU\_FILL);  lla=gluNewQuadric();  gluQuadricDrawStyle(lla, GLU\_FILL);  ls=gluNewQuadric();  gluQuadricDrawStyle(ls, GLU\_FILL);  us=gluNewQuadric();  gluQuadricDrawStyle(us, GLU\_FILL);  rua=gluNewQuadric();  gluQuadricDrawStyle(rua, GLU\_FILL);  rla=gluNewQuadric();  gluQuadricDrawStyle(rla, GLU\_FILL);  lul=gluNewQuadric();  gluQuadricDrawStyle(lul, GLU\_FILL);  lll=gluNewQuadric();  gluQuadricDrawStyle(lll, GLU\_FILL);  rul=gluNewQuadric();  gluQuadricDrawStyle(rul, GLU\_FILL);  rll=gluNewQuadric();  gluQuadricDrawStyle(rll, GLU\_FILL);  }  void main(int argc, char \*\*argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);  glutInitWindowSize(500, 500);  glutCreateWindow("robot");  myinit();  glutReshapeFunc(myReshape);  glutDisplayFunc(display);  glutMouseFunc(mouse);  glutCreateMenu(menu);  glutAddMenuEntry("torso", 0);  glutAddMenuEntry("head1", 1);  glutAddMenuEntry("head2", 2);  glutAddMenuEntry("right\_upper\_arm", 3);  glutAddMenuEntry("right\_lower\_arm", 4);  glutAddMenuEntry("left\_upper\_arm", 5);  glutAddMenuEntry("left\_lower\_arm", 6);  glutAddMenuEntry("right\_upper\_leg", 7);  glutAddMenuEntry("right\_lower\_leg", 8);  glutAddMenuEntry("left\_upper\_leg", 9);  glutAddMenuEntry("left\_lower\_leg", 10);  glutAddMenuEntry("quit", 11);  glutAttachMenu(GLUT\_MIDDLE\_BUTTON);  glutMainLoop();  } |

1. 應用說明：

利用類似於手臂的程式碼，將劍做出來，讓機器人看起來更帥。

十、參考資料：

Interactive Computer Graphics

<http://ivl.calit2.net/wiki/images/a/ad/17_ProceduralModeling.pdf>

https://cseweb.ucsd.edu//classes/wi18/cse167-a/lec15.pdf