計算機圖學單元介紹

一、英文主題：

Chapter 10: Modeling and Procedural Methods

二、中文主題：

單元 10：模組化與程序化

三、組別：

第 8 組

四、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；

五、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。

六、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();
}
```

七、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,
	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

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();
}
	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,
	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

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();}
	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);
	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

```
            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;
```

八、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();
```

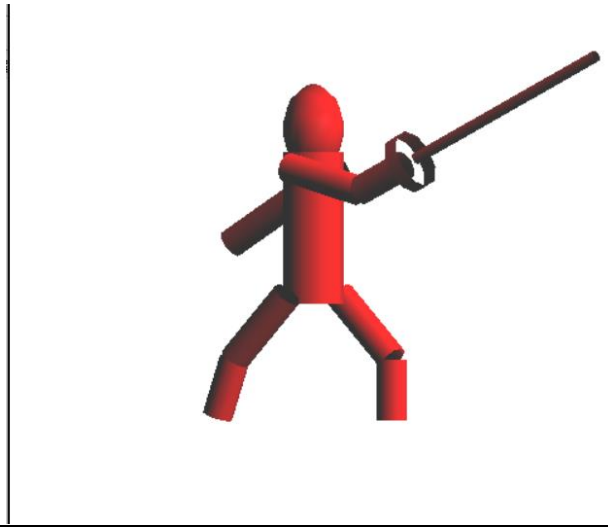
```
        gluQuadricDrawStyle(h, GLU_FILL);
        t=gluNewQuadric();
        gluQuadricDrawStyle(t, GLU_FILL);
        lua=gluNewQuadric();
        gluQuadricDrawStyle(lua, GLU_FILL);
        lla=gluNewQuadric();
        gluQuadricDrawStyle(lla, GLU_FILL);
        ls=gluNewQuadric();
        gluQuadricDrawStyle(ls, 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();
}
```



九、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