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 07 組

四、組員：

B0829001_姜念廷；B0829009_王凱心；B0829020_邵思絜；B0829021_游婷婷

B0829032_廖珮萱；B0829049_朱紫綸：B0829059_呂欣玲

五、作業分工：

(詳見作業報告)

六、功能簡述：

　　將完整的程式拆成較小的物件方便使用，並根據其相連部位做出相應的移動。將物體視為粒子集合，粒子系統中的個別粒子的動態行為可以用於模擬近似於物體的實際行為。

七、主要程式碼：

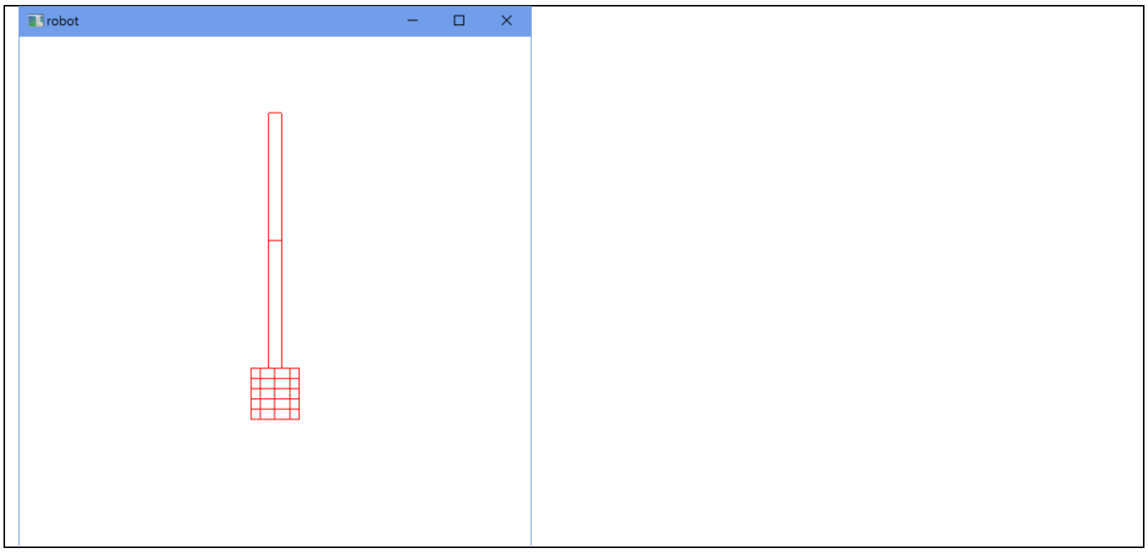
相關檔案：Ch_10_tm7_src1.cpp

```
#include <GL/glut.h>
#include <stdlib.h>

#define BASE_HEIGHT 2.0
#define BASE_RADIUS 1.0
#define LOWER_ARM_HEIGHT 5.0
#define LOWER_ARM_WIDTH 0.5
#define UPPER_ARM_HEIGHT 5.0
#define UPPER_ARM_WIDTH 0.5
```

```
typedef float point[3];

static GLfloat theta[] = {0.0,0.0,0.0};
static GLint axis = 0;
GLUquadricObj    *p; /* pointer to quadric object */

/* Define the three parts */
/* Note use of push/pop to return modelview matrix
to its state before functions were entered and use
rotation, translation, and scaling to create instances
of symbols (cube and cylinder */

void base()
{
     glPushMatrix();

/* rotate cylinder to align with y axis */

     glRotatef(-90.0, 1.0, 0.0, 0.0);

/* cyliner aligned with z axis, render with
     5 slices for base and 5 along length */

     gluCylinder(p, BASE_RADIUS, BASE_RADIUS, BASE_HEIGHT, 5, 5);
     glPopMatrix();
}

void upper_arm()
{
     glPushMatrix();
     glTranslatef(0.0, 0.5*UPPER_ARM_HEIGHT, 0.0);
     glScalef(UPPER_ARM_WIDTH, UPPER_ARM_HEIGHT, UPPER_ARM_WIDTH);
     glutWireCube(1.0);
     glPopMatrix();
}

void lower_arm()
{
     glPushMatrix();
     glTranslatef(0.0, 0.5*LOWER_ARM_HEIGHT, 0.0);
     glScalef(LOWER_ARM_WIDTH, LOWER_ARM_HEIGHT, LOWER_ARM_WIDTH);
     glutWireCube(1.0);
     glPopMatrix();
}

void display(void)
```

```
{

/* Accumulate ModelView Matrix as we traverse tree */

        glClear(GL_COLOR_BUFFER_BIT);
        glLoadIdentity();
        glColor3f(1.0, 0.0, 0.0);
        glRotatef(theta[0], 0.0, 1.0, 0.0);
        base();
        glTranslatef(0.0, BASE_HEIGHT, 0.0);
        glRotatef(theta[1], 0.0, 0.0, 1.0);
        lower_arm();
        glTranslatef(0.0, LOWER_ARM_HEIGHT, 0.0);
        glRotatef(theta[2], 0.0, 0.0, 1.0);
        upper_arm();
        glFlush();
        glutSwapBuffers();
}


void mouse(int btn, int state, int x, int y)
{

/* left button increase joint angle, right button decreases it */

if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
                {
                theta[axis] += 5.0;
                if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
                }
if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
                {
                theta[axis] -= 5.0;
                if( theta[axis] < 360.0 ) theta[axis] += 360.0;
                }
                display();
}

void menu(int id)
{

/* menu selects which angle to change or whether to quit */

      if(id == 1 ) axis=0;
      if(id == 2) axis=1;
      if(id == 3 ) axis=2;
      if(id ==4 ) exit(0);
```

```c
}

void
myReshape(int w, int h)
{
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	if (w <= h)
		glOrtho(-10.0, 10.0, -5.0 * (GLfloat) h / (GLfloat) w,
			15.0 * (GLfloat) h / (GLfloat) w, -10.0, 10.0);
	else
		glOrtho(-10.0 * (GLfloat) w / (GLfloat) h,
			10.0 * (GLfloat) w / (GLfloat) h, -5.0, 15.0, -10.0, 10.0);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

void myinit()
{
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 0.0, 0.0);

    p=gluNewQuadric(); /* allocate quadric object */
    gluQuadricDrawStyle(p, GLU_LINE); /* render it as wireframe */
}

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("base", 1);
	glutAddMenuEntry("lower arm", 2);
	glutAddMenuEntry("upper arm", 3);
	glutAddMenuEntry("quit", 4);
	glutAttachMenu(GLUT_MIDDLE_BUTTON);

	glutMainLoop();
}
```



八、程式說明：

1. 先使用#define 定義常數，以便解釋及更改。
2. 使用 glPushMatrix()和 glPopMatrix()回傳模組矩陣。
3. 把機器人手臂拆成底座、上手臂和下手臂三個函式，分別是 base()、upper_arm()和 lower_arm()。
4. 分別在各函數內設定旋轉、平移和縮放。
5. void myReshape()：改變視窗大小時，維持圖形比例，防止手臂旋轉的時候穿透。
6. void mouse()：設定滑鼠互動，左鍵向左旋轉 5 度，右鍵向右旋轉 5 度。
7. void menu()：透過選擇 1, 2, 3 或 4 來決定旋轉哪個部位或離開。
8. void display()：在尋訪樹時累積 ModelView 矩陣，呼叫各個函式把機器人手臂組裝起來。

九、延伸應用程式碼：

相關檔案：Ch_10_tm7_src2.cpp

```
#define NULL 0
#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;
typedef 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;

treenode torso_node, head_node, lua_node, rua_node, lll_node, rll_node,
lla_node, rla_node, rul_node, lul_node;


void traverse(treenode* 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_node);
```

```
	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_node.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_node.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_node.m);
break;

case 5 :
glLoadIdentity();
```

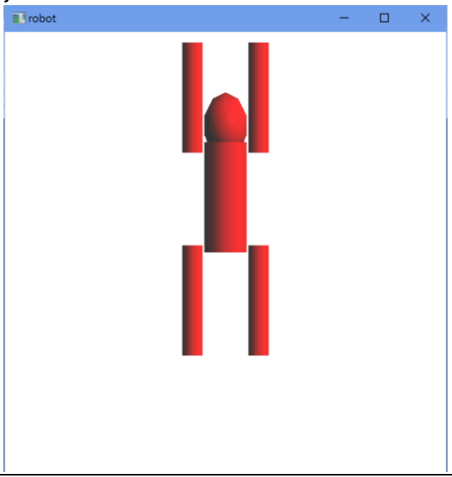
```
		glTranslatef(TORSO_RADIUS+UPPER_ARM_RADIUS, 0.9*TORSO_HEIGHT, 
0.0);
		glRotatef(theta[5], 1.0, 0.0, 0.0);
glGetFloatv(GL_MODELVIEW_MATRIX,rua_node.m);
break;

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_node.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_node.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_node.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_node.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_node.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_node.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 */

glLoadIdentity();
            glRotatef(theta[0], 0.0, 1.0, 0.0);
glGetFloatv(GL_MODELVIEW_MATRIX,torso_node.m);
torso_node.f = torso;
torso_node.sibling = NULL;
torso_node.child =    &head_node;

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_node.m);
head_node.f = head;
head_node.sibling = &lua_node;
head_node.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_node.m);
lua_node.f = left_upper_arm;
lua_node.sibling =   &rua_node;
lua_node.child = &lla_node;

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_node.m);
rua_node.f = right_upper_arm;
rua_node.sibling =   &lul_node;
rua_node.child = &rla_node;

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_node.m);
lul_node.f = left_upper_leg;
lul_node.sibling =   &rul_node;
lul_node.child = &lll_node;

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_node.m);
rul_node.f = right_upper_leg;
rul_node.sibling =   NULL;
rul_node.child = &rll_node;

glLoadIdentity();
            glTranslatef(0.0, UPPER_ARM_HEIGHT, 0.0);
            glRotatef(theta[4], 1.0, 0.0, 0.0);
```

```
glGetFloatv(GL_MODELVIEW_MATRIX,lla_node.m);
lla_node.f = left_lower_leg;
lla_node.sibling =   NULL;
lla_node.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_node.m);
rla_node.f = right_lower_arm;
rla_node.sibling =   NULL;
rla_node.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_node.m);
lll_node.f = left_lower_leg;
lll_node.sibling =   NULL;
lll_node.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_node.m);
rll_node.f = right_lower_leg;
rll_node.sibling =   NULL;
rll_node.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();
}
```



十、應用說明：

1. treenode()：設定左子右兄弟樹的節點。
2. traverse()：設定尋訪順序。
3. (1)left_upper_arm:建立左上手臂的物件

   (2)right_upper_arm():建立右上手臂的物件

   (3) left_lower_arm():建立左下手臂的物件

   (4) right_lower_arm():建立右下手臂的物件

   (5) left_upper_leg():建立左上腿的物件

   (6) left_lower_leg():建立左下腿的物件

   (7) right_upper_leg():建立右上腿的物件

   (8) right_lower_leg():建立右下腿的物件

4. myinit()用於將所有模組化的物件組合，將機器人拆成頭、軀幹、左上手臂、左下手臂、右上手臂、右下手臂、左上腿、左下腿、右上腿及右下腿 11 個部位。
5. gluCylinderGLUquadric *qobj, GLdouble baseRadius, GLdouble topRadius, GLdouble height, GLint slices, GLint stacks)，繪製圓柱，參數分別代表:

   (1) qobj：Quadric 物件 (使用 gluNewQuadric()) 建立。

   (2) baseRadius：位於 Z = 0 之圓柱的半徑。

   (3) topRadius：以 Z 高度為圓柱的半徑 。

   (4) height (高度)：圓柱的高度。

   (5) slices：圍繞 Z 軸的細分數目。

   (6) stacks：沿著 Z 軸的子分割數目。

6. mouse()：設定滑鼠互動及變換角度，分成 0~10 共 11 種 case，並根據不同 case 旋轉不同部位。
7. menu()：設定旋轉的 case。
8. myinit()：設定顏色、材質、光線、陰影，調配繪圖樣式至二次曲面，建立左子右兄弟樹。

十一、注意事項：

1. 相依關係表達出模組之間的耦合度強弱。關聯關係依其結合的程度(Degree of Coupling)而分為不同形式的關聯關係。
2. 若模組之間耦合度高，當參與關係的兩個模組彼此依靠，則無法單獨存在；若耦合度低，則模組可以獨立存在，但是如果沒有另一個物件的幫忙，某些模組會無法運作。若耦合度為 0，則兩個模組之間互不溝通，也沒有分享參數或資料。

十二、參考資料：

1. INTERACTIVE COMPUTER GRAPHICS A TOP-DOWN APPROACH WITH SHADER-BASED OPENGL 6th
   https://inspirit.net.in/books/academic/Interactive%20Computer%20Graphics.pdf