109(上)電腦圖學作業三機器人大變身

資工 4 甲

406261688

陳君平

程式架構 ：

- 程式初始設定 ：
  - glutInit(&argc, argv);
  - glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
  - glutInitWindowPosition(100, 100);
  - glutInitWindowSize(canvasWidth, canvasHeight);
  - glutCreateWindow("HW3");
  - init();
- 選單 ：
  - glutCreateMenu(menu);
  - glutAddMenuEntry("重置機器人", 0);
  - glutAddMenuEntry("打招呼", 1);
  - glutAddMenuEntry("仰臥起坐", 2);
  - glutAddMenuEntry("跑", 3);
  - glutAddMenuEntry("跳跳跳", 4);
  - glutAddMenuEntry("伏地挺身", 5);
  - glutAddMenuEntry("跳舞", 6);

- ■ glutAddMenuEntry("Quit", 9);
- ■ glutAttachMenu(GLUT_RIGHT_BUTTON);

● 顯示、按鍵、滑鼠……控制

- ■ glutReshapeFunc(reshape);
- ■ glutDisplayFunc(display);
- ■ glutMouseFunc(mouseButton);
- ■ glutMotionFunc(mouseMotion);
- ■ glutKeyboardFunc(keyboard);
- ■ glutIdleFunc(action);

與 402040347_張志泓 差異 ：

1. 人物模型修改
2. 人物顏色與背景顏色
3. 人物初始動作修改
4. 綜合重置功能
5. 新增跳舞動作
6. 新增伏地挺身動作

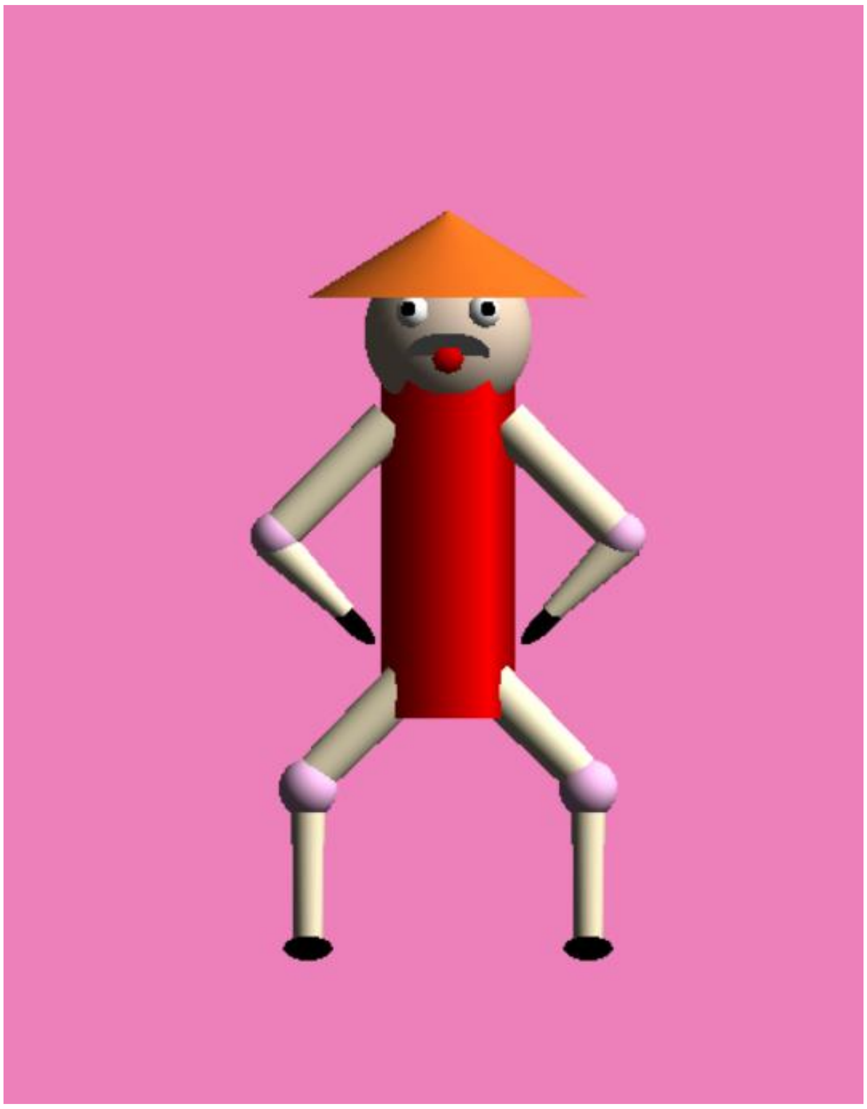
討論 ：

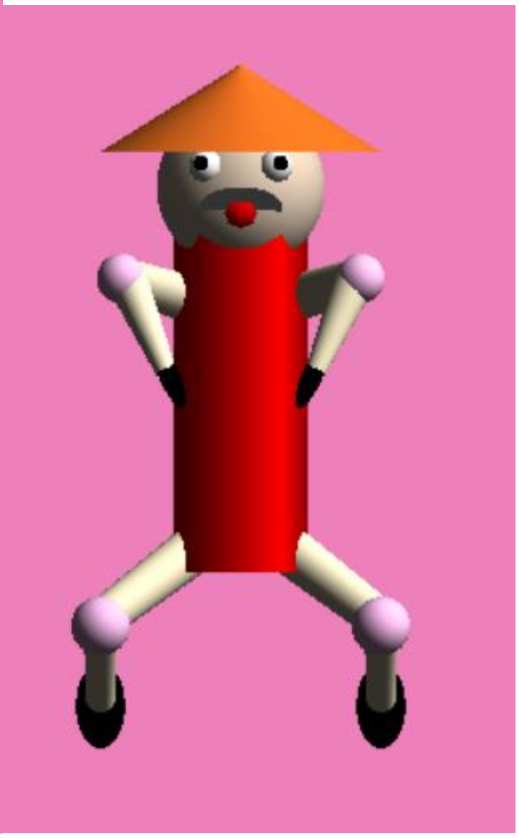
在本次作業中充分了解如何建構機器人，並了解對於機器人設定一些動作、模型所需的 function，而這次

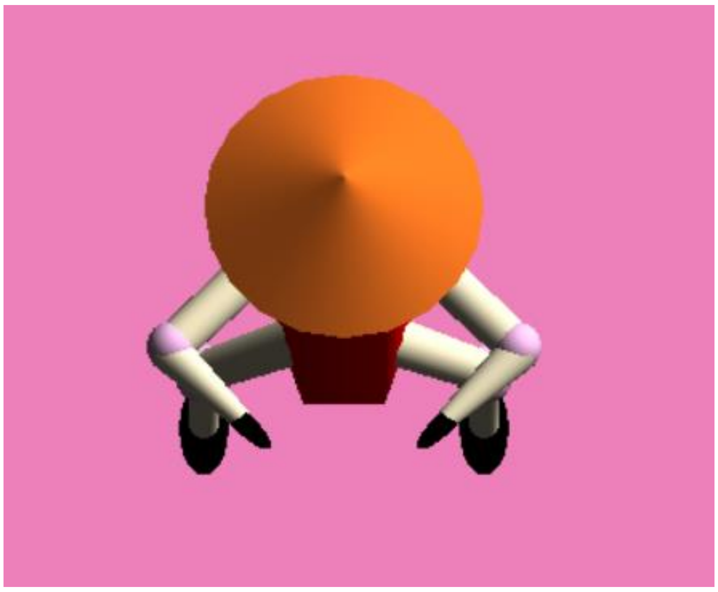
作業最有趣的部份是模型建構，雖然花了最多時間，

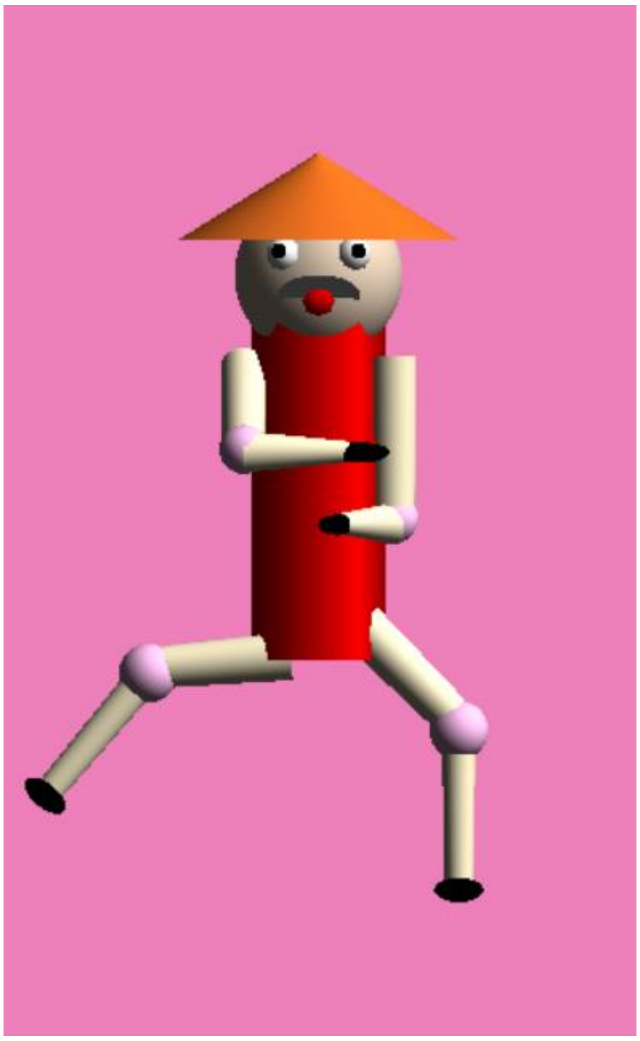
但可以作出令自己滿意的機器人。

執行畫面 ：









程式碼 ：

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

#include <stdlib.h>

#include <math.h>

#include <time.h>




#define PI 3.14159265358979323846f
```

```
#define TORSO_RADIUS 1.0

#define HEAD_RADIUS 1.0

#define UPPER_ARM_RADIUS 0.3

#define LOWER_ARM_RADIUS 0.3

#define UPPER_LEG_RADIUS 0.5

#define LOWER_LEG_RADIUS 0.4

#define JOINT_RADIUS 0.7


#define HEAD_HEIGHT 1.0

#define TORSO_HEIGHT 5.0

#define UPPER_ARM_HEIGHT 1.0

#define LOWER_ARM_HEIGHT 1.0

#define UPPER_LEG_HEIGHT 1.5

#define LOWER_LEG_HEIGHT 2.0



GLint canvasWidth = 700, canvasHeight = 800;
GLint mouseX, mouseY;

GLint actionNum = 0;


GLfloat init_Pos[3] = { -0.5, 5.0, 0.0 };

GLfloat init_Rot[3] = { 0.0, 0.0, 0.0 };

//分別以 xyz 軸旋轉之角度

GLfloat torsoRotate[3] = { 0.0, 0.0, 0.0 };

GLfloat robotRotate[3] = { 0.0, 0.0, 0.0 };

GLfloat headRotate[3] = { 0.0, 0.0, 0.0 };

GLfloat bodyRotate[3] = { 0.0, 0.0, 0.0 };

GLfloat leftUpArmRotate[3] = { 0.0, 0.0, -45.0 };

GLfloat leftLowArmRotate[3] = { 0.0, 0.0, 0.0 };

GLfloat rightUpArmRotate[3] = { 0.0, 0.0, 45.0 };

GLfloat rightLowArmRotate[3] = { 0.0, 0.0, 0.0 };

GLfloat leftUpLegRotate[3] = { 0.0, 0.0, 0.0 };

GLfloat leftLowLegRotate[3] = { 0.0, 0.0, 0.0 };

GLfloat rightUpLegRotate[3] = { 0.0, 0.0, 0.0 };

GLfloat rightLowLegRotate[3] = { 0.0, 0.0, 0.0 };
```

```
GLUquadricObj* cylinder, * sphere, * particialDisk, *
disk;

void init(void)
{
    GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
    GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 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);

    /*eye[0] = init_Pos[0] + eyeDistance * sin(yrotate) *
cos(xrotate);
    eye[2] = init_Pos[2] + eyeDistance * sin(yrotate) *
sin(xrotate);
```

```
    eye[1] = init_Pos[1] + eyeDistance * cos(yrotate);*/

    glShadeModel(GL_SMOOTH);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glDepthFunc(GL_LEQUAL);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_COLOR_MATERIAL); //重要!讓材質有
顏色!

    glClearColor(0.93, 1.0, 0.93, 1.0);

    cylinder = gluNewQuadric();
    gluQuadricDrawStyle(cylinder, GLU_FILL);
    sphere = gluNewQuadric();
    gluQuadricDrawStyle(sphere, GLU_FILL);
    particialDisk = gluNewQuadric();
    gluQuadricDrawStyle(particialDisk, GLU_FILL);
    disk = gluNewQuadric();
    gluQuadricDrawStyle(disk, GLU_FILL);


}


void reshape(int w, int h)
{
    glViewport(0, 0, w, h);


    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    //gluPerspective( 60,
(GLfloat)canvasWidth/(GLfloat)canvasHeight, 0.1,
1000 );
    if (w <= h)
        glOrtho(-10.0, 10.0, -5.0 * (GLfloat)h / (GLfloat)w,
            15.0 * (GLfloat)h / (GLfloat)w, -20.0, 20.0);
    else
        glOrtho(-10.0 * (GLfloat)w / (GLfloat)h,
            10.0 * (GLfloat)w / (GLfloat)h, -5.0, 15.0, -
20.0, 20.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    //gluLookAt( eye[0],eye[1],eye[2],at[0],at[1],at[2],u
p[0],up[1],up[2] );

}

void RotateObj(float* arr)
{
    glRotatef(*arr, 1.0, 0.0, 0.0);
    glRotatef(*(arr + 1), 0.0, 1.0, 0.0);
    glRotatef(*(arr + 2), 0.0, 0.0, 1.0);
}

void ActionChange(float* arr, float x, float y, float z)
{
    *arr = x;
    *(arr + 1) = y;
    *(arr + 2) = z;
}

void drawTorso()
{
    RotateObj(torsoRotate);
    glPushMatrix();
    //我是軀幹
    glColor3f(1.0, 0.0, 0.0);
    glRotatef(-90.0, 1.0, 0.0, 0.0);
    gluCylinder(cylinder, TORSO_RADIUS,
TORSO_RADIUS, TORSO_HEIGHT, 5, 5);
    glPopMatrix();

    glPopMatrix();
}

void drawHead()
```

```
{
    glPushMatrix();
    glTranslatef(0.0, TORSO_HEIGHT + HEAD_HEIGHT /
2.0, 0.0);
    RotateObj(headRotate);

    //臉
    glPushMatrix();
    glColor3f(1.0, 0.9, 0.8);
    glScalef(1.2, 1.2, 1.2);
    gluSphere(sphere, HEAD_RADIUS, 30, 30);
    glPopMatrix();

    //左眼
    glPushMatrix();
    glTranslatef(0.5 * HEAD_RADIUS, 0.3 *
HEAD_RADIUS,    HEAD_RADIUS);
    glColor3f(1.0, 1.0, 1.0);
    gluSphere(sphere, HEAD_RADIUS * 0.25, 10, 10);
    glPopMatrix();

    //左眼珠
    glPushMatrix();
    glTranslatef(0.55 * HEAD_RADIUS, 0.33 *
HEAD_RADIUS, 1.15*HEAD_RADIUS);
    glColor3f(0.0, 0.0, 0.0);
    gluSphere(sphere, HEAD_RADIUS * 0.13, 10, 10);
    glPopMatrix();

    //右眼
    glPushMatrix();
    glTranslatef(-0.5 * HEAD_RADIUS, 0.3 *
HEAD_RADIUS, HEAD_RADIUS);
    glColor3f(1.0, 1.0, 1.0);
    gluSphere(sphere, HEAD_RADIUS * 0.25, 10, 10);
    glPopMatrix();

    //右眼珠
    glPushMatrix();
    glTranslatef(-0.55 * HEAD_RADIUS, 0.33 *
HEAD_RADIUS, 1.15 * HEAD_RADIUS);
    glColor3f(0.0, 0.0, 0.0);
    gluSphere(sphere, HEAD_RADIUS * 0.13, 10, 10);
    glPopMatrix();

    //鬍子
    glPushMatrix();
    glTranslatef(0.0, -0.3 * HEAD_RADIUS,
1.2*HEAD_RADIUS);
    glRotatef(-180.0, 1.0, 0.0, 0.0);
    glColor3f(1.0, 1.0, 1.0);
    glScalef(2, 1, 1);
    gluPartialDisk(particialDisk, 0.1, 0.3, 20, 20, 80,
200); //殘缺圓盤(obj,內半徑,外半徑,slice,同心圓數,起
始角度位置,欲畫角度)
    glPopMatrix();
    //嘴巴
    glPushMatrix();
    glTranslatef(0.0, -0.3 * HEAD_RADIUS,
HEAD_RADIUS);
    glColor3f(1.0, 0.0, 0.0);
    glScalef(1.5, 1.5, 1);
    gluSphere(sphere, HEAD_RADIUS * 0.2, 10, 10);
    glPopMatrix();

    //帽子
    glPushMatrix();
    glColor3f(0.961, 0.455, 0.129);
    glTranslatef(0.0, 0.5 * HEAD_RADIUS, 0);
    glRotatef(-90.0, 1.0, 0.0, 0.0);
    glutSolidCone(2, 1.25, 20, 20);
    glColor3f(0.984, 0.682, 0.157);
    glPopMatrix();

    glPopMatrix();
```

```
}

void drawRightArm()
{
    glPushMatrix();
    glTranslatef(TORSO_RADIUS + 0.1 * JOINT_RADIUS,
0.8 * TORSO_HEIGHT, 0.0);
    RotateObj(rightUpArmRotate);

    //上臂
    glColor3f(1.0, 0.95, 0.8);
    glTranslatef(0.0, 0.5 * JOINT_RADIUS, 0.0);
    glPushMatrix();
    glScalef(1.0, 2.0, 1.0);
    glRotatef(90.0, 1.0, 0.0, 0.0);
    gluCylinder(cylinder, 1.1*UPPER_ARM_RADIUS,
UPPER_ARM_RADIUS, 1.2*UPPER_ARM_HEIGHT, 20,
20);
    glPopMatrix();
    //肘關節
    glColor3f(1.0, 0.75, 0.95);
    glTranslatef(0.0, -UPPER_ARM_HEIGHT - 1.9*
JOINT_RADIUS, 0.0);
    gluSphere(sphere, 0.5 * JOINT_RADIUS, 20, 20);

    RotateObj(rightLowArmRotate);
    //下臂
    glColor3f(1.0, 0.95, 0.8);
    glTranslatef(0.0, -0.1 * JOINT_RADIUS, 0.0);
    glPushMatrix();
    glRotatef(90.0, 1.0, 0.0, 0.0);
    glRotatef(-90.0, 0.0, 1.0, 0.0);
    gluCylinder(cylinder, LOWER_ARM_RADIUS, 0.5 *
LOWER_ARM_RADIUS, 1.5*LOWER_ARM_HEIGHT, 20,
20);
    glPopMatrix();
    //手掌
    glColor3f(0.0, 0.0, 0.0);
    glTranslatef(-1.6, -LOWER_ARM_HEIGHT +
1.4*JOINT_RADIUS, 0.0);
    glRotatef(180.0, 1.0, 0.0, 0.0);
    glRotatef(90.0, 0.0, 0.0, 1.0);
    glScalef(0.3, 1, 0.3);
    gluSphere(sphere, 0.7 * JOINT_RADIUS, 20, 20);

    glPopMatrix();
}

void drawLeftArm()
{
    glPushMatrix();
    glTranslatef(-TORSO_RADIUS - 0.1 * JOINT_RADIUS,
0.8 * TORSO_HEIGHT, 0.0);
    RotateObj(leftUpArmRotate);
    //上臂
    glColor3f(1.0, 0.95, 0.8);
    glTranslatef(0.0, 0.5 * JOINT_RADIUS, 0.0);
    glPushMatrix();
    glScalef(1.0, 2.0, 1.0);
    glRotatef(90.0, 1.0, 0.0, 0.0);
    gluCylinder(cylinder, 1.1 * UPPER_ARM_RADIUS,
UPPER_ARM_RADIUS, 1.2 * UPPER_ARM_HEIGHT, 20,
20);
    glPopMatrix();

    //肘關節
    glColor3f(1.0, 0.75, 0.95);
    glTranslatef(0.0, -UPPER_ARM_HEIGHT - 1.9 *
JOINT_RADIUS, 0.0);
    gluSphere(sphere, 0.5 * JOINT_RADIUS, 20, 20);

    RotateObj(leftLowArmRotate);
    //下臂
    glColor3f(1.0, 0.95, 0.8);
    glTranslatef(0.0, -0.1 * JOINT_RADIUS, 0.0);
    glPushMatrix();
    glRotatef(90.0, 1.0, 0.0, 0.0);
    glRotatef(90.0, 0.0, 1.0, 0.0);
    gluCylinder(cylinder, LOWER_ARM_RADIUS, 0.5 *
LOWER_ARM_RADIUS, 1.5 * LOWER_ARM_HEIGHT, 20,
20);
    glPopMatrix();

    //手掌
    glColor3f(0.0, 0.0, 0.0);
    glTranslatef(1.6, -LOWER_ARM_HEIGHT + 1.4 *
JOINT_RADIUS, 0.0);
    glRotatef(180.0, 1.0, 0.0, 0.0);
    glRotatef(90.0, 0.0, 0.0, 1.0);
    glScalef(0.3, 1, 0.3);
    gluSphere(sphere, 0.7 * JOINT_RADIUS, 20, 20);
    glPopMatrix();
}

void drawRightLeg()
{
    glPushMatrix();
    glTranslatef(1.1 * TORSO_RADIUS, 0.0, 0.0);
    RotateObj(rightUpLegRotate);
    glColor3f(1.0, 0.95, 0.8);

    //上大腿
    glTranslatef(-0.5, 0.5, 0.0);
    glPushMatrix();
    glRotatef(90.0, 1.0, 0.0, 0.0);
    glRotatef(45.0, 0.0, 1.0, 0.0);
    gluCylinder(cylinder, 0.7*UPPER_LEG_RADIUS, 0.6 *
UPPER_LEG_RADIUS, 1.2*UPPER_LEG_HEIGHT, 30, 30);
    glPopMatrix();
    //膝蓋關節
    glColor3f(1.0, 0.75, 0.95);
    glTranslatef(1.4, -UPPER_LEG_HEIGHT, 0.0);
    gluSphere(sphere, 0.6 * JOINT_RADIUS, 30, 30);

    RotateObj(rightLowLegRotate);
    //下大腿
    glColor3f(1.0, 0.95, 0.8);
    glTranslatef(0.0, -0.4 * JOINT_RADIUS, 0.0);
    glPushMatrix();
    glRotatef(90.0, 1.0, 0.0, 0.0);
    gluCylinder(cylinder, 0.6*LOWER_LEG_RADIUS, 0.5*
LOWER_LEG_RADIUS, LOWER_LEG_HEIGHT, 30, 30);
    glPopMatrix();

    //腳裸
    glColor3f(0.0, 0.0, 0.0);
    glTranslatef(0.0, -LOWER_LEG_HEIGHT, 0.0);
    glScalef(0.5, 0.25, 1);
    gluSphere(sphere, JOINT_RADIUS, 30, 30);
    glPopMatrix();
}

void drawLeftLeg()
{
    glPushMatrix();
    glTranslatef(-1.1 * TORSO_RADIUS, 0.0, 0.0);
    RotateObj(leftUpLegRotate);
    glColor3f(1.0, 0.95, 0.8);

    //上大腿
    glTranslatef(0.5, 0.5, 0.0);
    glPushMatrix();
    glRotatef(90.0, 1.0, 0.0, 0.0);
    glRotatef(-45.0, 0.0, 1.0, 0.0);
    gluCylinder(cylinder, 0.7 * UPPER_LEG_RADIUS, 0.6
* UPPER_LEG_RADIUS, 1.2 * UPPER_LEG_HEIGHT, 30,
30);
```

```
    glPopMatrix();

    //膝蓋關節
    glColor3f(1.0, 0.75, 0.95);
    glTranslatef(-1.4, -UPPER_LEG_HEIGHT, 0.0);
    gluSphere(sphere, 0.6 * JOINT_RADIUS, 30, 30);

    RotateObj(leftLowLegRotate);
    //下大腿
    glColor3f(1.0, 0.95, 0.8);
    glTranslatef(0.0, -0.4 * JOINT_RADIUS, 0.0);
    glPushMatrix();
    glRotatef(90.0, 1.0, 0.0, 0.0);
    gluCylinder(cylinder, 0.6 * LOWER_LEG_RADIUS, 0.5
* LOWER_LEG_RADIUS, LOWER_LEG_HEIGHT, 30, 30);
    glPopMatrix();

    //腳裸
    glColor3f(0.0, 0.0, 0.0);
    glTranslatef(0.0, -LOWER_LEG_HEIGHT, 0.0);
    glScalef(0.5, 0.25, 1);
    gluSphere(sphere, JOINT_RADIUS, 30, 30);

    glPopMatrix();
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glClearColor(0.93, 0.5, 0.73, 1.0);

    //初始化位置
    glTranslatef(init_Pos[0], init_Pos[1], init_Pos[2]);
    //視角
    RotateObj(init_Rot);
    //旋轉機器人
    RotateObj(robotRotate);

    //構築機器人
    drawTorso();
    drawHead();
    drawRightArm();
    drawLeftArm();
    drawRightLeg();
    drawLeftLeg();

    glFlush();
    glutSwapBuffers();
}


void mouseButton(int button, int state, int x, int y)
{
    if (state == GLUT_DOWN)
        if (button == GLUT_LEFT_BUTTON)
        {
            mouseX = x;
            mouseY = y;
        }
}

void mouseMotion(int x, int y)
{
    if (x > mouseX && y > mouseY)
    {
        init_Rot[1] += 1.0;
        if (init_Rot[1] > 360.0) init_Rot[1] -= 360.0;
        init_Rot[0] += 1.0;
        if (init_Rot[0] > 360.0) init_Rot[0] -= 360.0;
    }

    if (x > mouseX && y < mouseY)
    {
        init_Rot[1] += 1.0;
        if (init_Rot[1] > 360.0) init_Rot[1] -= 360.0;
        init_Rot[0] -= 1.0;
        if (init_Rot[0] < 0.0) init_Rot[0] += 360.0;
    }

    if (x < mouseX && y > mouseY)
    {
        init_Rot[1] -= 1.0;
        if (init_Rot[1] < 0.0) init_Rot[1] += 360.0;
        init_Rot[0] += 1.0;
        if (init_Rot[0] > 360.0) init_Rot[0] -= 360.0;
    }

    if (x < mouseX && y < mouseY)
    {
        init_Rot[1] -= 1.0;
        if (init_Rot[1] < 0.0) init_Rot[1] += 360.0;
        init_Rot[0] -= 1.0;
        if (init_Rot[0] < 0.0) init_Rot[0] += 360.0;
```

```
}


if (x == mouseX && y > mouseY)
{
     init_Rot[0] += 1.0;
     if (init_Rot[0] > 360.0) init_Rot[0] -= 360.0;
}
if (x == mouseX && y < mouseY)
{
     init_Rot[0] -= 1.0;
     if (init_Rot[0] < 0.0) init_Rot[0] += 360.0;
}
if (y == mouseY && x > mouseX)
{
     init_Rot[1] += 1.0;
     if (init_Rot[1] > 360.0) init_Rot[1] -= 360.0;
}
if (y == mouseY && x < mouseX)
{
        init_Rot[1] -= 1.0;

        if (init_Rot[1] < 0.0) init_Rot[1] += 360.0;

    }


    glutPostRedisplay();

}


void keyboard(unsigned char key, int x, int y)

{

    switch (key)

    {

    case 'w':

        init_Pos[1] += 0.5;

        glutPostRedisplay();

        break;

    case 's':

        init_Pos[1] -= 0.5;

        glutPostRedisplay();

        break;
    case 'a':
        init_Pos[0] -= 0.5;
        glutPostRedisplay();
        break;
    case 'd':
        init_Pos[0] += 0.5;
        glutPostRedisplay();
        break;
    default:
        break;
    }
}

void reSetRobot()
{
    ActionChange(torsoRotate, 0.0, 0.0, 0.0);
    ActionChange(robotRotate, 0.0, 0.0, 0.0);
    ActionChange(headRotate, 0.0, 0.0, 0.0);
    ActionChange(leftUpArmRotate, 0.0, 0.0, -45.0);
    ActionChange(leftLowArmRotate, 0.0, 0.0, 0.0);

    ActionChange(rightUpArmRotate, 0.0, 0.0, 45.0);

    ActionChange(rightLowArmRotate, 0.0, 0.0, 0.0);

    ActionChange(leftUpLegRotate, 0.0, 0.0, 0.0);

    ActionChange(leftLowLegRotate, 0.0, 0.0, 0.0);

    ActionChange(rightUpLegRotate, 0.0, 0.0, 0.0);

    ActionChange(rightLowLegRotate, 0.0, 0.0, 0.0);

}


void menu(int id)

{

    switch (id)

    {

    case 0:

        init_Rot[0] = 0.0;

        init_Rot[1] = 0.0;

        init_Rot[2] = 0.0;

        init_Pos[0] = -0.5;

        init_Pos[1] = 5.0;
        init_Pos[2] = 0.0;
        reSetRobot();
        actionNum = 0;
        glutPostRedisplay();
        break;
    case 1:
        reSetRobot();
        actionNum = 1;
        glutPostRedisplay();
        break;
    case 2:
        reSetRobot();
        actionNum = 2;
        glutPostRedisplay();
        break;
    case 3:
        reSetRobot();
        actionNum = 3;
        glutPostRedisplay();
        break;
    case 4:
        reSetRobot();
        actionNum = 4;
        glutPostRedisplay();
        break;
    case 5:
        reSetRobot();
        actionNum = 5;
        glutPostRedisplay();
        break;
    case 6:
        reSetRobot();
        actionNum = 6;
        glutPostRedisplay();
        break;
    case 9:
        exit(0);
        break;
        default:
            break;
        }
    }

    void run(int time)
    {
        switch (time % 4)
        {
        case 0:
            ActionChange(leftUpArmRotate, -60.0, 0.0, -
    40.0);
            ActionChange(leftLowArmRotate, -65.0, 0.0,
    40.0);
            ActionChange(rightUpArmRotate, 60.0, 0.0,
    40.0);
            ActionChange(rightLowArmRotate, -65.0, 0.0, -
    20.0);
```

```
        ActionChange(leftUpLegRotate, 60.0, 0.0, 0.0);
        ActionChange(leftLowLegRotate, 40.0, 0.0, 0.0);
        ActionChange(rightUpLegRotate, -60.0, 0.0, 0.0);
        ActionChange(rightLowLegRotate, 40.0, 0.0, 0.0);
        break;
    case 1:
        ActionChange(leftUpArmRotate, 0.0, 0.0, -40.0);
        ActionChange(leftLowArmRotate, -65.0, 0.0,
40.0);
        ActionChange(rightUpArmRotate, 0.0, 0.0, 40.0);
        ActionChange(rightLowArmRotate, -65.0, 0.0, -
20.0);

        ActionChange(leftUpLegRotate, 0.0, 0.0, 0.0);
        ActionChange(leftLowLegRotate, 10.0, 0.0, 0.0);
        ActionChange(rightUpLegRotate, 0.0, 0.0, 0.0);
        ActionChange(rightLowLegRotate, 10.0, 0.0, 0.0);
        break;
    case 2:
        ActionChange(leftUpArmRotate, 60.0, 0.0, -40.0);
        ActionChange(leftLowArmRotate, -65.0, 0.0,
40.0);
        ActionChange(rightUpArmRotate, -60.0, 0.0,
40.0);
        ActionChange(rightLowArmRotate, -65.0, 0.0, -
20.0);

        ActionChange(leftUpLegRotate, -60.0, 0.0, 0.0);
        ActionChange(leftLowLegRotate, 40.0, 0.0, 0.0);
        ActionChange(rightUpLegRotate, 60.0, 0.0, 0.0);
        ActionChange(rightLowLegRotate, 40.0, 0.0, 0.0);
        break;
    case 3:
        ActionChange(leftUpArmRotate, 0.0, 0.0, -40.0);
        ActionChange(leftLowArmRotate, -65.0, 0.0,
40.0);
        ActionChange(rightUpArmRotate, 0.0, 0.0, 40.0);
        ActionChange(rightLowArmRotate, -65.0, 0.0, -
20.0);

          ActionChange(leftUpLegRotate, 0.0, 0.0, 0.0);
          ActionChange(leftLowLegRotate, 10.0, 0.0, 0.0);
          ActionChange(rightUpLegRotate, 0.0, 0.0, 0.0);
          ActionChange(rightLowLegRotate, 10.0, 0.0, 0.0);
          break;
     default:
          break;
     }

}

void pushUp(int time)
{
     ActionChange(robotRotate, 70.0, 0.0, 0.0);
     switch (time % 2)
     {
     case 0:
        ActionChange(headRotate, 0.0, 0.0, 0.0);
        ActionChange(torsoRotate, 0.0, 0.0, 0.0);

        ActionChange(leftUpArmRotate, -90.0, 0.0, -
45.0);
        ActionChange(rightUpArmRotate, -90.0, 0.0,
45.0);

        ActionChange(leftLowArmRotate, 0.0, 0.0, 0.0);
        ActionChange(rightLowArmRotate, 0.0, 0.0, 0.0);
        break;
    case 1:
        ActionChange(headRotate, 0.0, 0.0, 0.0);
        ActionChange(torsoRotate, 0.0, 0.0, 0.0);

        ActionChange(leftUpArmRotate, -90.0, 0.0, 0.0);
        ActionChange(rightUpArmRotate, -90.0, 0.0, 0.0);

        ActionChange(leftLowArmRotate, 0.0, 0.0, -90.0);
        ActionChange(rightLowArmRotate, 0.0, 0.0,
90.0);
        break;

    default:
        break;

    }
}

void sitUps(int time)
{
    ActionChange(robotRotate, -70.0, 0.0, 0.0);
    switch (time % 2)
    {
    case 0:
        ActionChange(headRotate, 0.0, 0.0, 0.0);
        ActionChange(torsoRotate, 0.0, 0.0, 0.0);
        ActionChange(leftUpArmRotate, -60.0, 0.0, 0.0);
        ActionChange(rightUpArmRotate, -60.0, 0.0, 0.0);

        ActionChange(leftUpLegRotate, -40.0, 0.0, 0.0);
        ActionChange(leftLowLegRotate, 100.0, 0.0, 0.0);
        ActionChange(rightUpLegRotate, -40.0, 0.0, 0.0);
        ActionChange(rightLowLegRotate, 100.0, 0.0,
0.0);
        break;
    case 1:
        ActionChange(headRotate, 20.0, 0.0, 0.0);
        ActionChange(torsoRotate, 20.0, 0.0, 0.0);

        ActionChange(leftUpArmRotate, -40.0, 0.0, 0.0);
        ActionChange(rightUpArmRotate, -40.0, 0.0, 0.0);

        ActionChange(leftUpLegRotate, -60.0, 0.0, 0.0);
        ActionChange(leftLowLegRotate, 120.0, 0.0, 0.0);
        ActionChange(rightUpLegRotate, -60.0, 0.0, 0.0);
```

```
            ActionChange(rightLowLegRotate, 120.0, 0.0,
0.0);
            break;

      default:
            break;

      }
}

void happyJump(int time)
{
      switch (time % 2)
      {
      case 0:
            init_Pos[1] = 5.0;

            ActionChange(leftUpArmRotate, -60.0, 60.0, -
40.0);
```

```
        ActionChange(leftLowArmRotate, -120.0, 0.0,
0.0);
        ActionChange(rightUpArmRotate, -60.0, -60.0,
40.0);
        ActionChange(rightLowArmRotate, -120.0, 0.0,
0.0);

        ActionChange(leftUpLegRotate, -40.0, 0.0, 0.0);
        ActionChange(leftLowLegRotate, 100.0, 0.0, 0.0);
        ActionChange(rightUpLegRotate, -40.0, 0.0, 0.0);
        ActionChange(rightLowLegRotate, 100.0, 0.0,
0.0);

        break;
    case 1:
        init_Pos[1] = 8.0;

        ActionChange(leftUpArmRotate, -60.0, 0.0, 0.0);
        ActionChange(leftLowArmRotate, -90.0, 0.0, -
30.0);
          ActionChange(rightUpArmRotate, -60.0, 0.0, 0.0);
          ActionChange(rightLowArmRotate, -90.0, 0.0,
30.0);

          ActionChange(rightUpLegRotate, 0.0, 0.0, 30.0);
          ActionChange(rightLowLegRotate, 0.0, 0.0, 0.0);
          ActionChange(leftUpLegRotate, 0.0, 0.0, -30.0);
          ActionChange(leftLowLegRotate, 0.0, 0.0, 0.0);

          break;

     default:
          break;
     }

}

void dance(int time)
```

```
{
    switch (time % 2)
    {
    case 0:

        ActionChange(leftUpArmRotate, -60.0, 0.0, 0.0);
        ActionChange(rightUpArmRotate, 0.0, 60.0, 0.0);

        ActionChange(leftUpLegRotate, 0.0, 0.0, -40.0);
        ActionChange(rightUpLegRotate, 00.0, 0.0, 0.0);

        break;
    case 1:
        ActionChange(rightUpArmRotate, -60.0, 0.0, 0.0);
        ActionChange(leftUpArmRotate, 0.0, 60.0, 0.0);

        ActionChange(leftUpLegRotate, 0.0, 0.0, 0.0);
        ActionChange(rightUpLegRotate, 00.0, 0.0, 40.0);
        break;
    default:
        break;
    }
}
void action()
{
    long time_box;
    time_box = time(0);
    switch (actionNum)
    {
    case 1:
        ActionChange(leftUpArmRotate, 0.0, 0.0, -90.0);
        ActionChange(leftLowArmRotate, 0.0, 0.0,
135.0);
        ActionChange(rightUpArmRotate, 0.0, 0.0, 90.0);
        ActionChange(rightLowArmRotate, 0.0, 0.0, -
45.0);
        break;
```

```
case 2:
    sitUps(time_box);
    break;
case 3:
    run(time_box);
    break;
case 4:
    happyJump(time_box);
    break;
case 5:
    pushUp(time_box);
    break;
case 6:
    dance(time_box);
    break;
default:
    break;
}
glutPostRedisplay();
```

```
}



int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE |
GLUT_DEPTH);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(canvasWidth, canvasHeight);
    glutCreateWindow("HW3");

    init();

    glutCreateMenu(menu);
    glutAddMenuEntry("重置機器人", 0);
    glutAddMenuEntry("打招呼", 1);
    glutAddMenuEntry("仰臥起坐", 2);
    glutAddMenuEntry("跑", 3);
    glutAddMenuEntry("跳跳跳", 4);
    glutAddMenuEntry("伏地挺身", 5);
    glutAddMenuEntry("跳舞", 6);
    glutAddMenuEntry("Quit", 9);
    glutAttachMenu(GLUT_RIGHT_BUTTON);

    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutMouseFunc(mouseButton);
    glutMotionFunc(mouseMotion);
    glutKeyboardFunc(keyboard);
    glutIdleFunc(action);
    //glutTimerFunc(300,timerMove,1);
    glutMainLoop();

    return 0;
}
```