作業三　機器人大變身

405262180

資工三乙

劉育騏

討論:本次作業，知道許多新的參數，函式用法，計時器等等之類的。還有物件，畫 3D 圖形，以及一些動作基本旋轉，放大，位移設定。以及許許多多東西，蠻有趣的。獲益良多。

程式架構:

部位關係

身體->頭部

身體->左手->左下臂->手->手 2

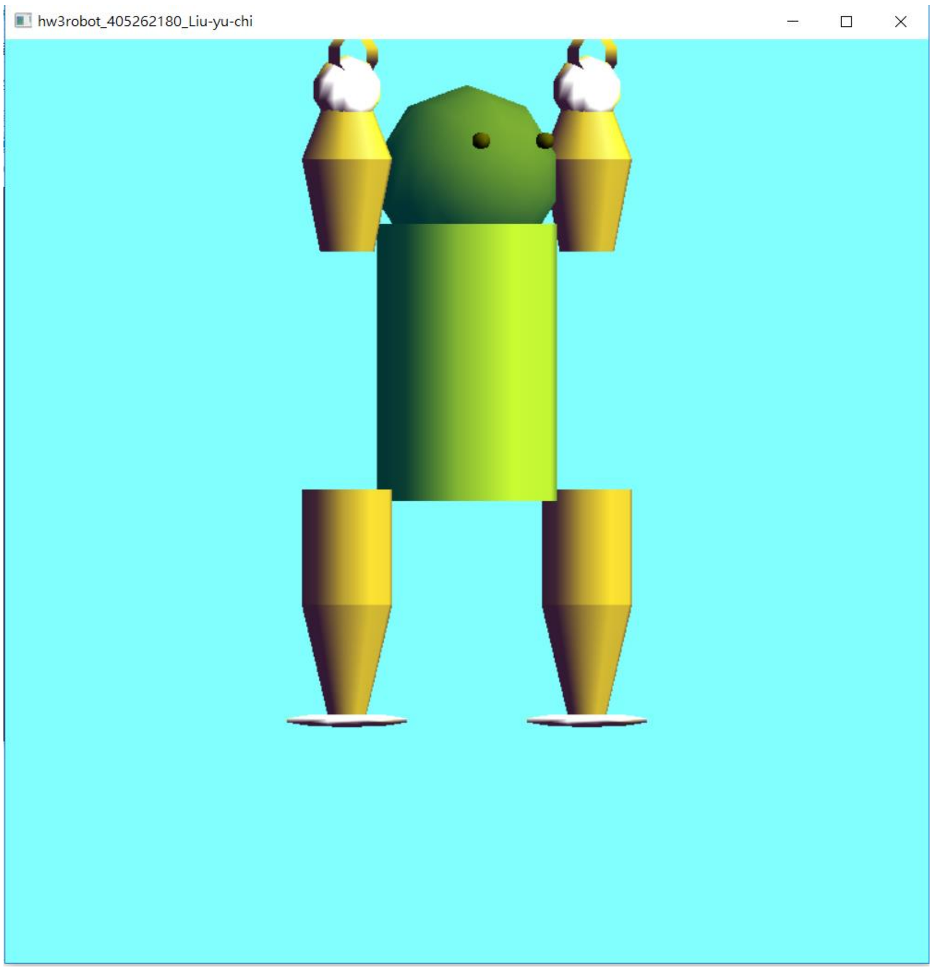
身體->右手->右下臂->手->手 2

身體->左腳->左下腳->腳掌

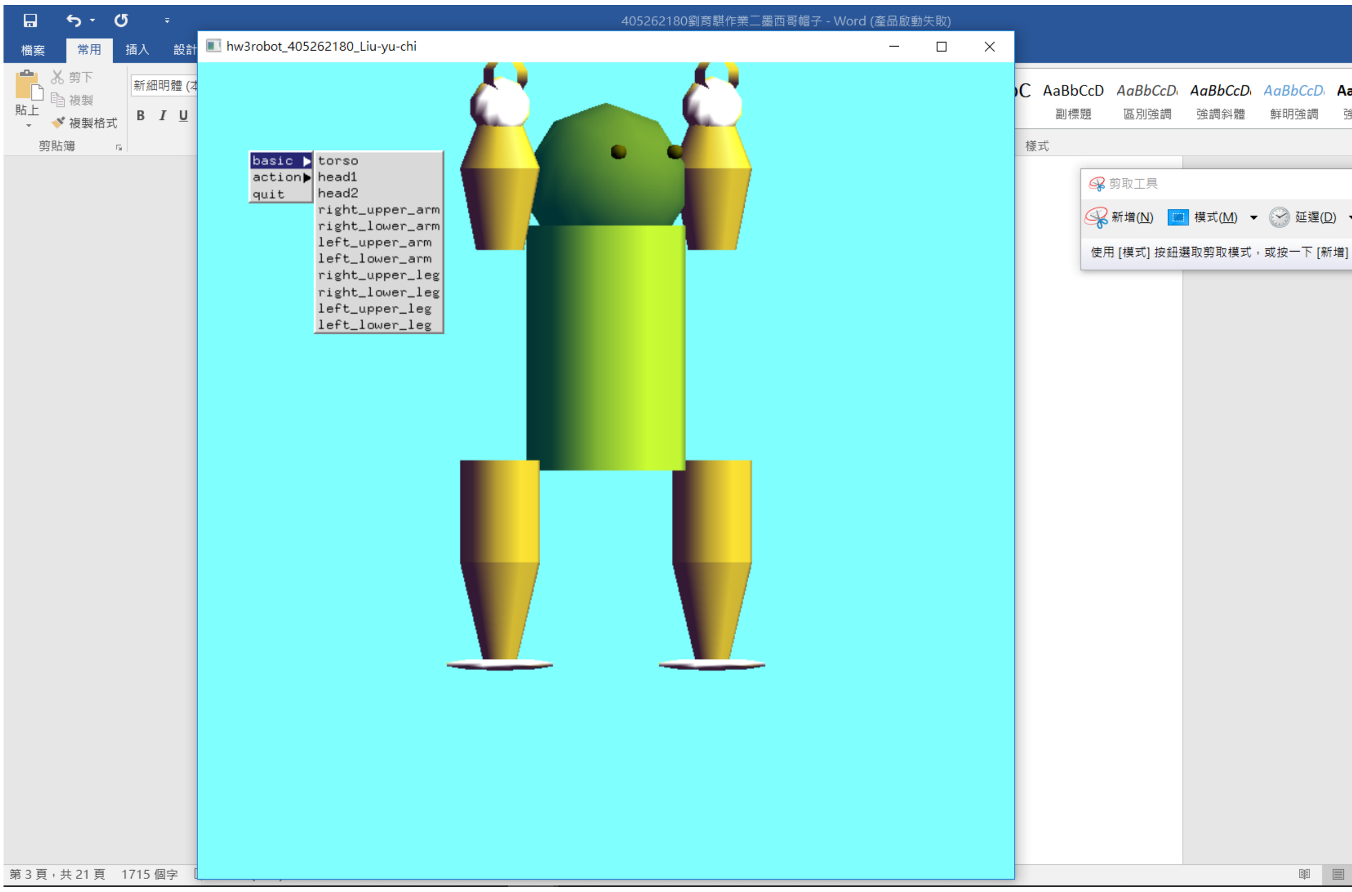
身體->右腳->右下腳->腳掌

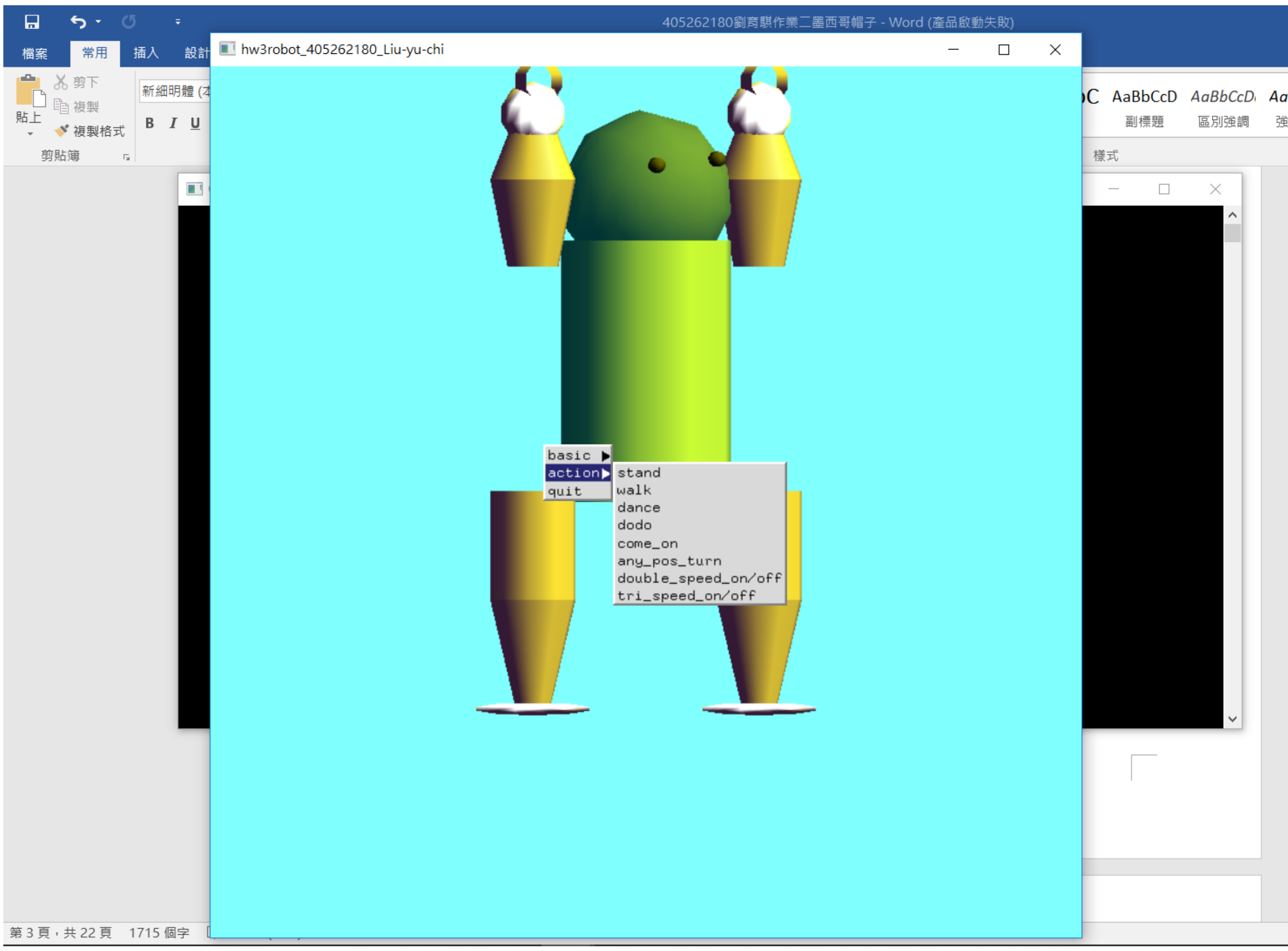
執行畫面:

初始:



操作選項:


405262180劉育騏作業二圖西哥帽子 - Word (產品啟動失敗)
hw3robot_405262180_Liu-yu-chi
basic
action
quit
torso
head1
head2
right_upper_arm
right_lower_arm
left_upper_arm
left_lower_arm
right_upper_leg
right_lower_leg
left_upper_leg
left_lower_leg
第 3 頁，共 21 頁　1715 個字



程式碼

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif
#define TORSO_HEIGHT 6
#define TORSO_RADIUS 2
#define UPPER_ARM_HEIGHT 2.0
#define LOWER_ARM_HEIGHT 1.5
#define UPPER_ARM_RADIUS    1
#define LOWER_ARM_RADIUS    1
#define UPPER_LEG_RADIUS    1
```

```
#define LOWER_LEG_RADIUS    1
#define LOWER_LEG_HEIGHT 2.5
#define UPPER_LEG_HEIGHT 2.5
#define HEAD_HEIGHT 2
#define HEAD_RADIUS 2

static GLfloat theta[11] = {30.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 180.0, 0.0, 180.0, 0.0}; //身體部位旋轉
static GLfloat atheta[8] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0}; //  對z軸旋轉
static GLint angle = 2; //index
const GLfloat rot = 5.0;// rotate angle
GLUquadricObj *t, *h, *lua, *lla, *rua, *rla, *lll, *rll, *rul, *lul;
int time_clock = 0;
int state = 0; //機器人狀態
int mills = 22; //倒數時間
GLboolean isDoubleSpeed = false;
GLboolean isThirdSpeed = false;

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);

    glPushMatrix();
    glColor3f(0.0, 0.0, 0.0);
    glTranslatef(0.4, 0.4, 1.0);
    gluSphere(h,0.1,10,3);
    glPopMatrix();
    glPushMatrix();
    glColor3f(0.0, 0.0, 0.0);
    glTranslatef(-0.4, 0.4, 1.0);
    gluSphere(h,0.1,10,3);
    glPopMatrix();

    glPopMatrix();
}

void left_upper_arm()
{
    glPushMatrix();
    glRotatef(-90.0, 1.0, 0.0, 0.0);
    glColor3f(1, 0.5, 1);
    gluCylinder(lua,UPPER_ARM_RADIUS*0.6, 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*0.4,
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*0.6, 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*0.4,
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*0.4,
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*0.4,
LOWER_LEG_HEIGHT,10,10);
    glPopMatrix();
}
void left_hand()
{
    glPushMatrix();
    glScalef(0.5, 0.5, 0.5);
    gluSphere(h,1.5,10,10);
    glPopMatrix();
}
void left_hand2()
{
    glPushMatrix();
    glScalef(0.5, 0.5, 0.5);
    gluCylinder(t, 1, 1, 1, 10, 10);
    glPopMatrix();
}
void right_hand()
{
    glPushMatrix();
    glScalef(0.5, 0.5, 0.5);
    gluSphere(h,1.5,10,10);
    glPopMatrix();
}
void right_hand2()
{
    glPushMatrix();
    glScalef(0.5, 0.5, 0.5);
    gluCylinder(t, 1, 1, 1, 10, 10);
    glPopMatrix();
}
void left_foot()
{
    glPushMatrix();
    glScalef(1, 0.1, 0.5);
    gluSphere(h,1.5,10,10);
    glPopMatrix();
}
void right_foot()
{
    glPushMatrix();
    glScalef(1, 0.1, 0.5);
    gluSphere(h,1.5,10,10);
    glPopMatrix();
}


void
display(void)
{
     glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
     glLoadIdentity();
     glColor3f(0, 1, 1);

     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, 0.0, 1.0);
     glTranslatef(0.0, -0.5*HEAD_HEIGHT, 0.0);
     head();
     glPopMatrix();

     glPushMatrix();
     glColor3f(0, 1, 1);

     glTranslatef(-(TORSO_RADIUS+UPPER_ARM_RADIUS), 0.9*TORSO_HEIGHT, 0.0);
     glRotatef(theta[3], 1.0, 0.0, 0.0);
     glRotatef(atheta[0], 0.0, 0.0, 1.0);
     left_upper_arm();

     glTranslatef(0.0, UPPER_ARM_HEIGHT, 0.0);
```

```
	glRotatef(theta[4], 1.0, 0.0, 0.0);
	glRotatef(atheta[1], 0.0, 0.0, 1.0);
	left_lower_arm();

	glTranslatef(0.0, LOWER_ARM_HEIGHT, 0.0);
	left_hand();

	glTranslatef(0.0, LOWER_ARM_HEIGHT*0.5, 0.0);
	left_hand2();

	glPopMatrix();

	glPushMatrix();
	glTranslatef(TORSO_RADIUS+UPPER_ARM_RADIUS, 0.9*TORSO_HEIGHT, 0.0);
	glRotatef(theta[5], 1.0, 0.0, 0.0);
	glRotatef(atheta[2], 0.0, 0.0, 1.0);
	right_upper_arm();

	glTranslatef(0.0, UPPER_ARM_HEIGHT, 0.0);

	glRotatef(theta[6], 1.0, 0.0, 0.0);
	glRotatef(atheta[3], 0.0, 0.0, 1.0);
	right_lower_arm();

	glTranslatef(0.0, LOWER_ARM_HEIGHT, 0.0);
	right_hand();

	glTranslatef(0.0, LOWER_ARM_HEIGHT*0.5, 0.0);
	right_hand2();
	glPopMatrix();

	glPushMatrix();
	glTranslatef(-(TORSO_RADIUS+UPPER_LEG_RADIUS), 0.1*UPPER_LEG_HEIGHT,
0.0);
	glRotatef(theta[7], 1.0, 0.0, 0.0);
	glRotatef(atheta[4], 0.0, 0.0, 1.0);
	left_upper_leg();
	glTranslatef(0.0, UPPER_LEG_HEIGHT, 0.0);
	glRotatef(theta[8], 1.0, 0.0, 0.0);
	glRotatef(atheta[5], 0.0, 0.0, 1.0);
	left_lower_leg();

	glTranslatef(0.0, LOWER_LEG_HEIGHT, 0.0);
	left_foot();
	glPopMatrix();

	glPushMatrix();
	glTranslatef(TORSO_RADIUS+UPPER_LEG_RADIUS, 0.1*UPPER_LEG_HEIGHT, 0.0);
	glRotatef(theta[9], 1.0, 0.0, 0.0);
	glRotatef(atheta[6], 0.0, 0.0, 1.0);
	right_upper_leg();

	glTranslatef(0.0, UPPER_LEG_HEIGHT, 0.0);
	glRotatef(theta[10], 1.0, 0.0, 0.0);
	glRotatef(atheta[7], 0.0, 0.0, 1.0);
	right_lower_leg();

	glTranslatef(0.0, LOWER_LEG_HEIGHT, 0.0);
	right_foot();

	glPopMatrix();

	glFlush();
	glutSwapBuffers();
}

void recover(void)
{
	theta[0] = 30.0;
	theta[1] = 0.0;
	theta[2] = 0.0;
	theta[3] = 0.0;
	theta[4] = 0.0;
	theta[5] = 0.0;
```

```
        theta[6] = 0.0;
        theta[7] = 180.0;
        theta[8] = 0.0;
        theta[9] = 180.0;
        theta[10] = 0.0;
        for(int i = 0; i < 8; i++){
                atheta[i] = 0.0;
        }

  }

void midPunch(void)
{
      theta[0] = 0.0;
      theta[1] = 0.0;
      theta[2] = 0.0;
      theta[3] = 0.0;
      theta[4] = 0.0;
      theta[5] = 0.0;
      theta[6] = 0.0;
      theta[7] = 180.0;
      theta[8] = 0.0;
      theta[9] = 180.0;
      theta[10] = 0.0;
      atheta[0] = 90.0;
      atheta[1] = 0.0;
      atheta[2] = 0.0;
      atheta[3] = 90.0;
      atheta[4] = 10.0;
      atheta[5] = 100.0;
      atheta[6] = -90.0;
      atheta[7] = 0.0;
}

void leftPunch(void)
{
      theta[0] -= 6.0;
      theta[1] = 0.0;
```

```
    theta[2] = 0.0;
    theta[3] = 0.0;
    theta[4] = 0.0;
    theta[5] = 0.0;
    theta[6] = 0.0;
    theta[7] = 180.0;
    theta[8] = 0.0;
    theta[9] = 180.0;
    theta[10] = 0.0;
    atheta[0] = 90.0;
    atheta[1] = 0.0;
    atheta[2] = 0.0;
    atheta[3] = 10.0;
    atheta[4] = 90.0;
    atheta[5] = -90.0;
    atheta[6] = 10.0;
    atheta[7] = 0.0;
}

void rightPunch(void)
{
    theta[0] += 6.0;
    theta[1] = 0.0;
    theta[2] = 0.0;
    theta[3] = 0.0;
    theta[4] = 0.0;
    theta[5] = 0.0;
    theta[6] = 0.0;
    theta[7] = 180.0;
    theta[8] = 0.0;
    theta[9] = 180.0;
    theta[10] = 0.0;
    atheta[0] = 90.0;
    atheta[1] = -90.0;
    atheta[2] = -90.0;
    atheta[3] = 0.0;
    atheta[4] = 90.0;
    atheta[5] = 0.0;
    atheta[6] = -90.0;
    atheta[7] = 90.0;
}


void TimerFunction(int value)
{

        switch(state){
                case 1:/**走路時間**/
                        time_clock++;
                        if(time_clock < 10){
                                theta[0] = 30.0;
                                theta[1] = 0.0;
                                theta[2] = -10.0;
                                theta[3] = 230.0;
                                theta[4] = -30.0;
                                theta[5] = 140.0;
                                theta[6] = -30.0;
                                theta[7] = 100.0;
                                theta[8] =    70.0;
                                theta[9] = 180.0;
                                theta[10]= 0.0;
                                for(int i = 0 ; i < 8 ; i++)
                                        atheta[i] = 0.0;
                                }
                        else if(time_clock < 20){
                                theta[0] = 30.0;
                                theta[1] = 0.0;
                                theta[2] = 10.0;
                                theta[3] = 140.0;
                                theta[4] = -30.0;
                                theta[5] = 230.0;
                                theta[6] = -30.0;
                                theta[7] = 180.0;
                                theta[8] =      0.0;
                                theta[9] = 120.0;
                                theta[10] = 70.0;
```

```
            for(int i = 0 ; i < 8 ; i++)
                atheta[i] = 0.0;
        }
        else{
            time_clock = 0;
        }
        break;
case 2:/**跳舞時間**/
        time_clock++;
        if(time_clock < 10)
           rightPunch();
        else if(time_clock < 20)
           leftPunch();
        else if(time_clock < 30)
            midPunch();
        else
           time_clock = 0;
        break;
case 3:/**抖動**/
        time_clock++;
        if(time_clock < 10){
            theta[0] = 30.0;
            theta[1] = 0.0;
            theta[2] = -10.0;
            theta[3] = 230.0;
            theta[4] = -30.0;
            theta[5] = 140.0;
            theta[6] = -30.0;
            theta[7] = 100.0;
            theta[8] =    70.0;
            theta[9] = 180.0;
            theta[10]= 0.0;
            for(int i = 0 ; i < 8 ; i++)
                 atheta[i] = 30.0;
            }
        else if(time_clock < 20){
            theta[0] = 30.0;
            theta[1] = 0.0;
```

```
                theta[2] = 10.0;
                theta[3] = 140.0;
                theta[4] = -30.0;
                theta[5] = 230.0;
                theta[6] = -30.0;
                theta[7] = 180.0;
                theta[8] =      0.0;
                theta[9] = 120.0;
                theta[10] = 70.0;
                for(int i = 0 ; i < 8 ; i++)
                        atheta[i] = -30.0;
        }
        else{
                time_clock = 0;
        }
        break;
case 4:/**挑釁**/
        time_clock++;
        if(time_clock < 5){
                theta[2] += rot;
        }
        else if(time_clock < 5+18){
                theta[3] += rot;
                theta[5] += rot;
        }
        else if(time_clock < 10+18){
                theta[4] -= rot;
                theta[6] -= rot;
        }
        else if(time_clock < 15+18){
                theta[4] += rot;
                theta[6] += rot;
        }
        else if(time_clock < 20+18){
                theta[4] -= rot;
                theta[6] -= rot;
        }
        else if(time_clock < 25+18){
```

```
                        theta[4] += rot;
                        theta[6] += rot;
                    }
                    else{
                        time_clock = 0;
                        theta[2] = 0;
                        theta[3] = 0;
                        theta[5] = 0;
                        theta[6] = 0;
                    }
                    break;
                case 5:/**轉轉轉**/
                    time_clock++;
                    if(time_clock < 36){
                        theta[3] += 2*rot;
                        theta[5] -= 2*rot;
                    }
                    else if(time_clock < 36+9)
                        theta[0] += rot;
                    else{
                        time_clock = 0;
                    }
                    break;
        }
            glutPostRedisplay();
            glutTimerFunc(mills ,TimerFunction, 1);
  }

void mouse(int btn, int state, int x, int y)
{
        if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
            {
            theta[angle] += rot;
            if( theta[angle] > 360.0 ) theta[angle] -= 360.0;
            }
        if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
            {
            theta[angle] -= rot;
```

```
                if( theta[angle] < 360.0 ) theta[angle] += 360.0;
                }
        glutPostRedisplay();
}

void menu(int id)
{
        if(id == 2)
                exit(0);
}

void bas_menu(int id){
                angle=id;
}

void act_menu(int id){
        switch(id){
                case 0:
                        state = 0;
                        recover();
                        break;
                case 1:
                        state = 1;
                        break;
                case 2:
                        state = 2;
                        break;
                case 3:
                        state = 3;
                        break;
                case 4:
                        recover();
                        time_clock = 0;
                        state = 4;
                        break;
                case 5:
                        state = 5;
                break;
```

```
		case 6:
			isDoubleSpeed ^= 1;
			if(isDoubleSpeed){
				mills = 11;
				isThirdSpeed = false;
			}
			else
				mills = 22;
			break;
		case 7:
			isThirdSpeed ^= 1;
			if(isThirdSpeed){
				mills = 5;
				isDoubleSpeed = false;
			}
			else
				mills = 22;
			break;
	}
}


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, 1.0, 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);

        glColorMaterial(GL_FRONT_AND_BACK,GL_AMBIENT);
        glEnable(GL_COLOR_MATERIAL);
        glClearColor(0.5, 1, 1, 1);

/** 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);
}

int main(int argc, char **argv)
{
	int basic_menu, action_menu;
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
	glutInitWindowSize(800, 800);
	glutCreateWindow("hw3robot_405262180_Liu-yu-chi");
	myinit();

	glutReshapeFunc(myReshape);
	glutTimerFunc(mills, TimerFunction, 1);
	glutDisplayFunc(display);
	glutMouseFunc(mouse);

	basic_menu = glutCreateMenu(bas_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);


        action_menu = glutCreateMenu(act_menu);
        glutAddMenuEntry("stand", 0);
        glutAddMenuEntry("walk", 1);
        glutAddMenuEntry("dance", 2);
        glutAddMenuEntry("dodo", 3);
        glutAddMenuEntry("come_on", 4);
        glutAddMenuEntry("any_pos_turn", 5);
        glutAddMenuEntry("double_speed_on/off", 6);
        glutAddMenuEntry("tri_speed_on/off", 7);

        glutCreateMenu(menu);
        glutAddSubMenu("basic", basic_menu);
        glutAddSubMenu("action", action_menu);
        glutAddMenuEntry("quit", 2);
        glutAttachMenu(GLUT_MIDDLE_BUTTON);

        glEnable(GL_DEPTH_TEST);

        glutMainLoop();
}
```