**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

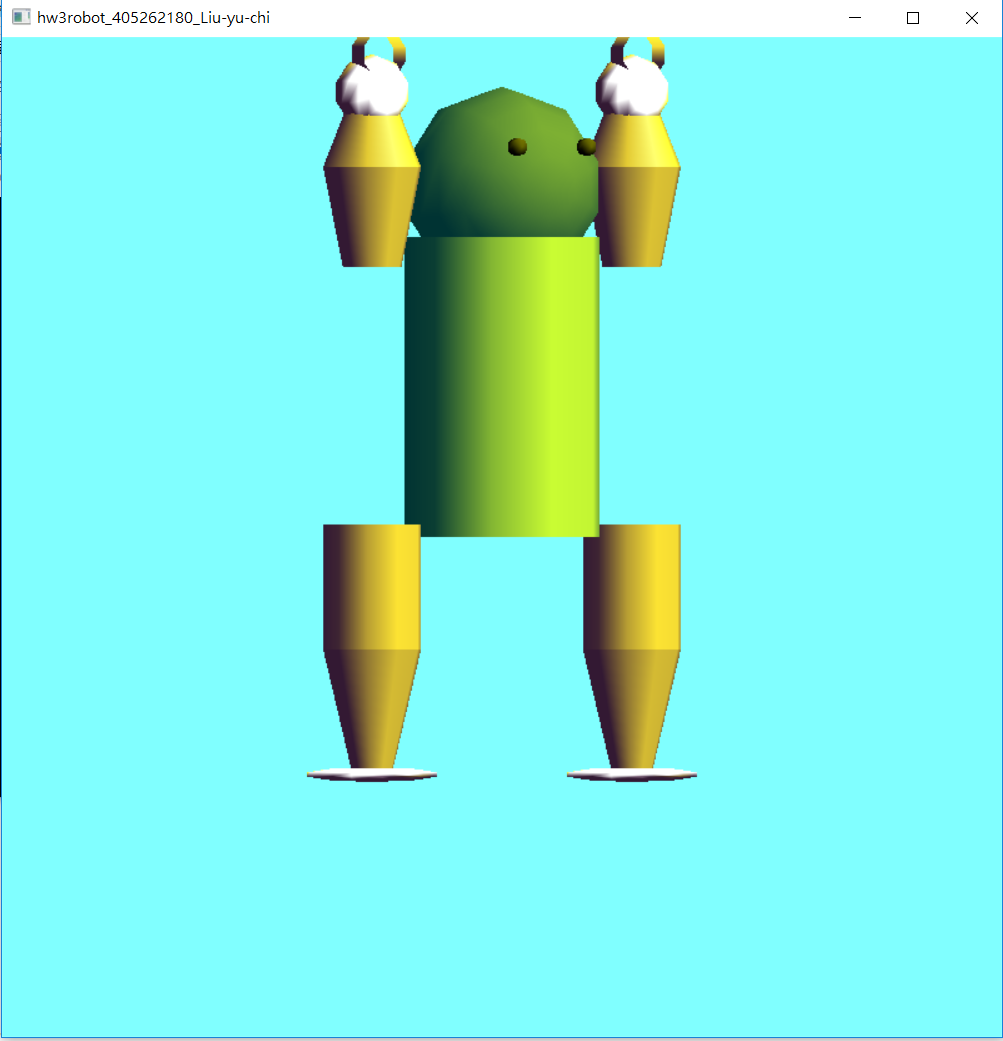
身體->右手->右下臂->手->手2

身體->左腳->左下腳->腳掌

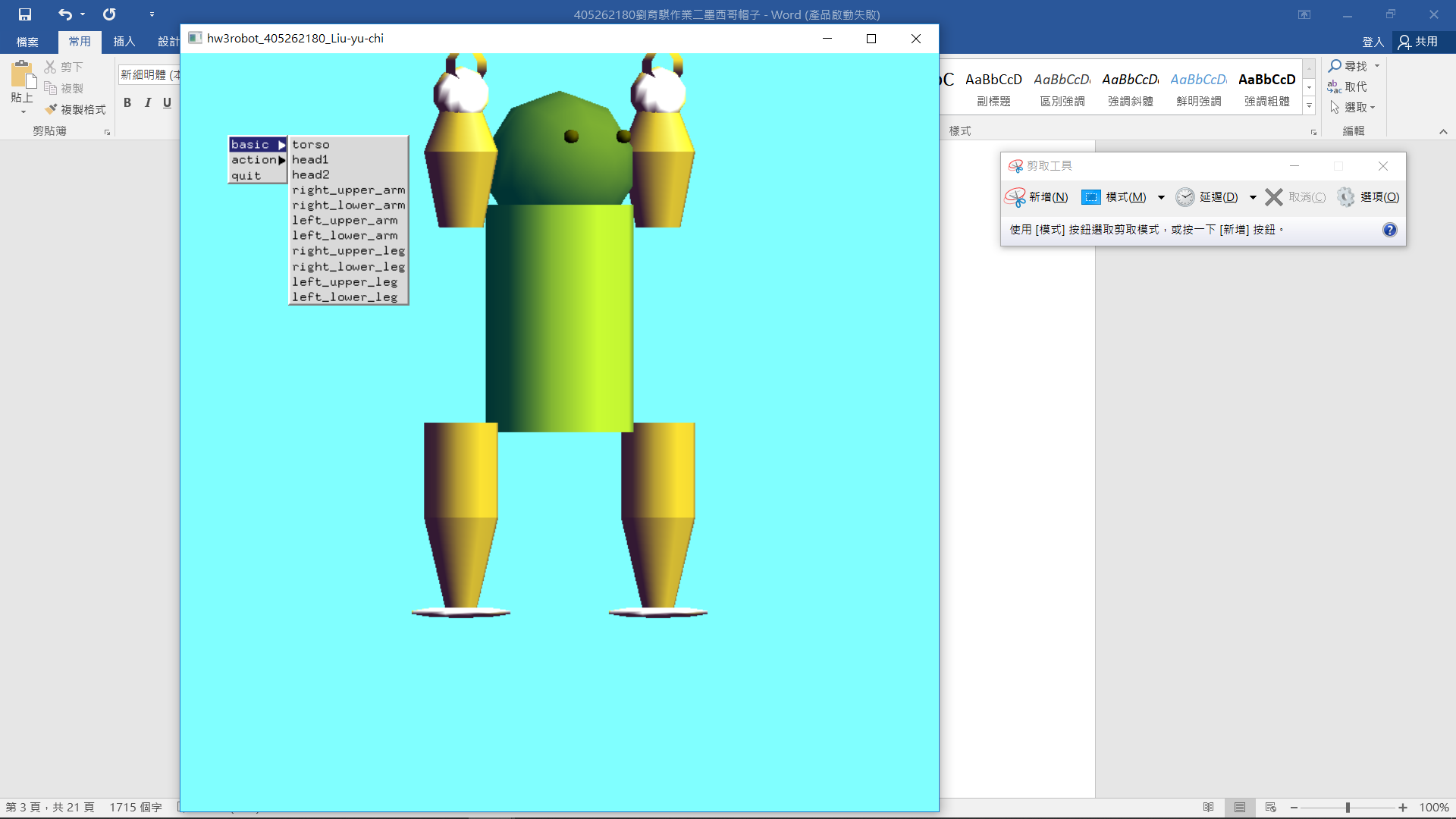
身體->右腳->右下腳->腳掌

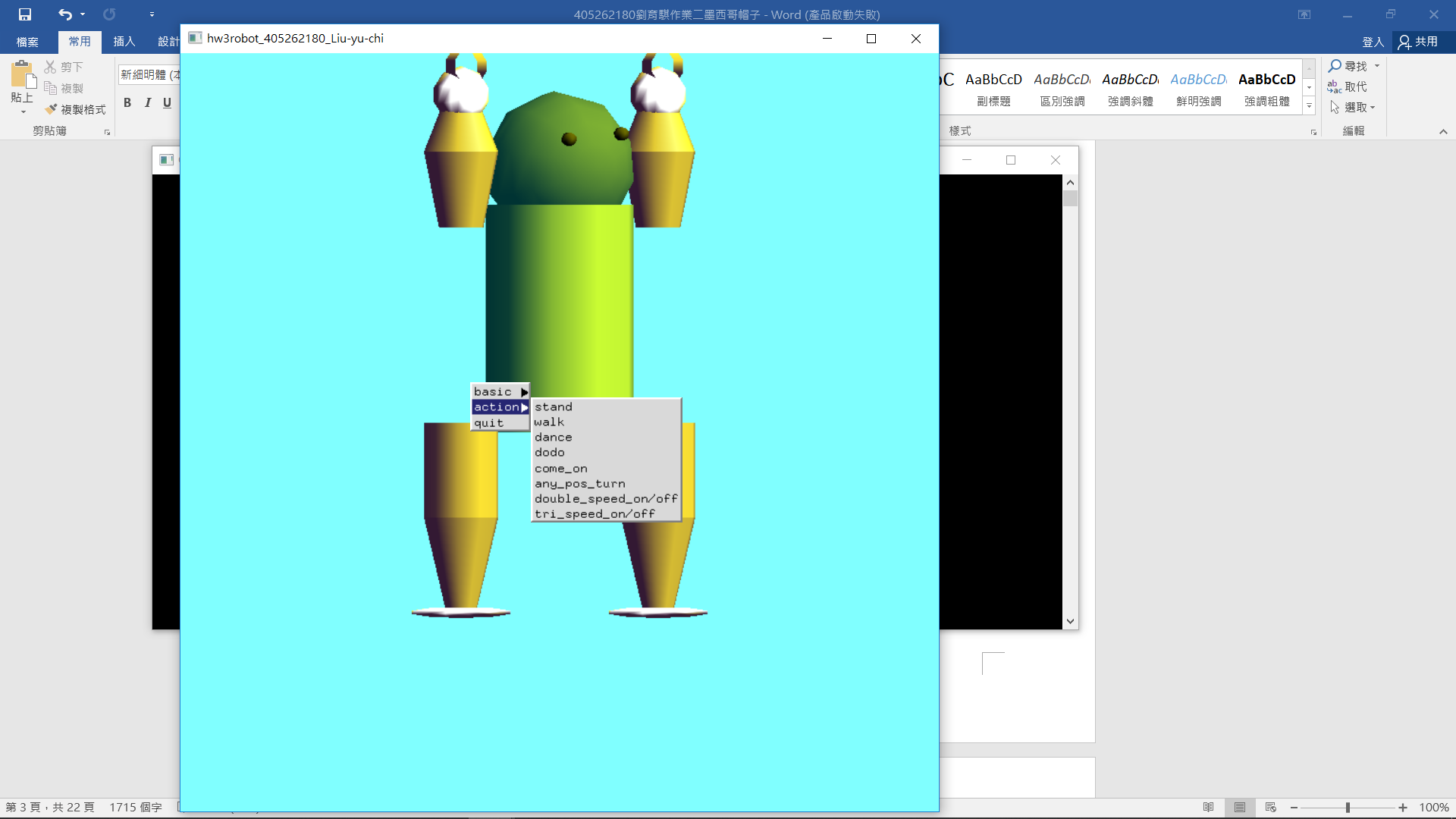
執行畫面:

初始:



操作選項:





**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();

}