406261597 資工三甲 林子傑

機器人大變身

架構

Display 函式會依序連接到 Torso, Head, RightArm, LeftArm, RightLeg, LeftLeg 來架構整個身體。

Menu 函式則是會依據右鍵選單更新 actionNum，再由 Action 執行相對應的程式，有 cheer, swim, dance, kick 四種，每種指令會根據 time_box 的值去執行動作。

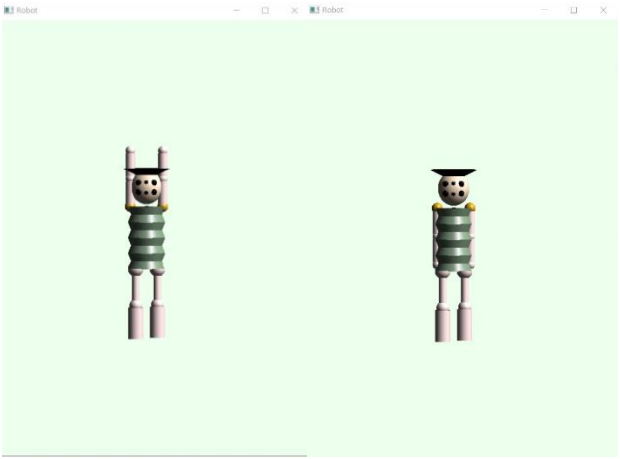
討論

這份作業讓我了解到 OpenGL 是怎麼建構出有"繼承"關係的物件，利用 glPushMatrix 來記錄父物件的狀態，當子物件建構好後，用 glPopMatrix 返回父物件，這樣一來既不會讓子物件破壞父物件的狀態，當父物件跟動時，子物件也能有相對應的更動。glRotatef 也在這次作業中扮演很重要的角色，必須設定旋轉角度，才能呈現出動作。我覺得這份作業知道要怎麼做以後，就變很好玩了，可以設計出自己想要的動作，有時侯會不小心調出非常滑稽的動作呢。

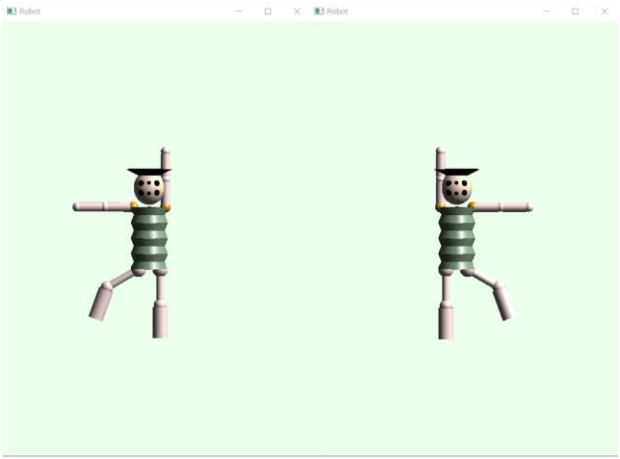
執行畫面

初始畫面

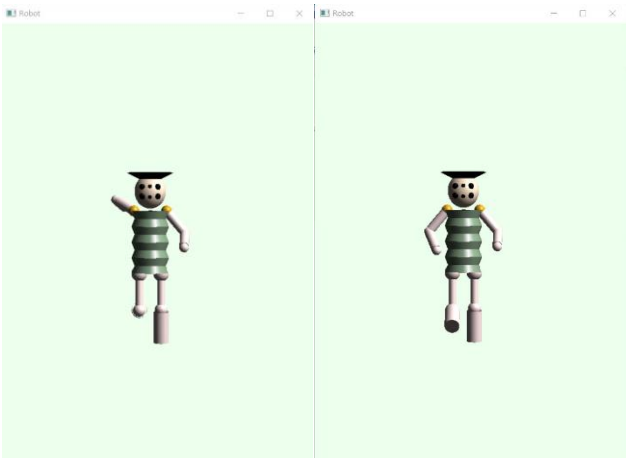
歡呼



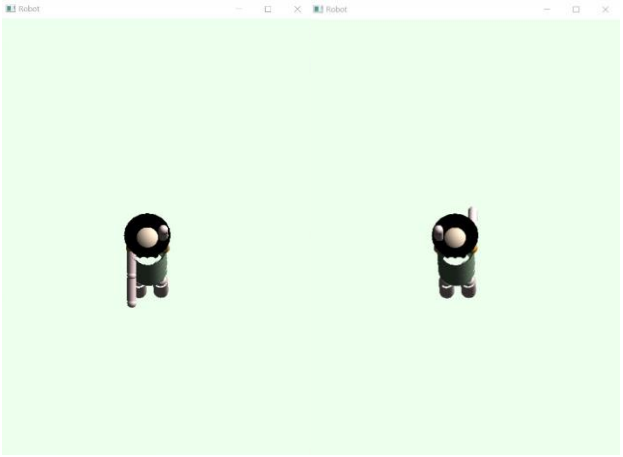
跳舞



踢



游泳



程式碼

```
#include <GL/glut.h>
#include <iostream>
#include <math.h>
#include <stdlib.h>
#include <time.h>

using namespace std;

#define PI 3.14159265358979323846f

#define TORSO_HEIGHT 5.0
#define UPPER_ARM_HEIGHT 1.5
#define LOWER_ARM_HEIGHT 1
#define UPPER_LEG_RADIUS 0.4
#define LOWER_LEG_RADIUS 0.4
#define LOWER_LEG_HEIGHT 2.0
#define UPPER_LEG_HEIGHT 2.0
#define TORSO_RADIUS 1.0
#define UPPER_ARM_RADIUS 0.3
#define LOWER_ARM_RADIUS 0.3
#define HEAD_HEIGHT 1.5
#define HEAD_RADIUS 1.0
#define JOINT_RADIUS 0.5

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};

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};
GLdouble CutUp[4] = {0.0, -1.0, 0.0, 0.0};
GLfloat bodyRotate[3] = {0.0, 0.0, 0.0};
GLfloat leftHair[3] = {0.0, 0.0, -30.0};
GLfloat rightHair[3] = {0.0, 0.0, 30.0};
```

```
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 *t, *h;

void init(void)
{
	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, 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);

	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);
	glEnable(GL_COLOR_MATERIAL);
```

```
	glClearColor(0.93, 1.0, 0.93, 1.0);

	t = gluNewQuadric();
	gluQuadricDrawStyle(t, GLU_FILL);
	h = gluNewQuadric();
	gluQuadricDrawStyle(h, GLU_FILL);
}

void reshape(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, -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();
}

void Rotate(float *p)
{
	glRotatef(*p, 1.0, 0.0, 0.0);
	glRotatef(*(p + 1), 0.0, 1.0, 0.0);
	glRotatef(*(p + 2), 0.0, 0.0, 1.0);
}

void Change(float *q, float x, float y, float z)
{
	*q = x;
	*(q + 1) = y;
	*(q + 2) = z;
```

```
}

void Torso()
{
	Rotate(torsoRotate);
	glPushMatrix();
	glColor3f(0.5, 0.6, 0.5);
	glRotatef(-90.0, 1.0, 0.0, 0.0);

	glTranslatef(0.0, 0.0, 0.7 * TORSO_HEIGHT);
	gluCylinder(t, TORSO_RADIUS, 1.2 * TORSO_RADIUS, 0.1 * TORSO_HEIGHT, 20,
			20);
	glTranslatef(0.0, 0.0, -0.1 * TORSO_HEIGHT);
	gluCylinder(t, 1.2 * TORSO_RADIUS, TORSO_RADIUS, 0.1 * TORSO_HEIGHT, 20,
			20);
	for (int i = 0; i < 3; ++i)
	{
		glTranslatef(0.0, 0.0, -0.1 * TORSO_HEIGHT);
		gluCylinder(t, TORSO_RADIUS, 1.2 * TORSO_RADIUS, 0.1 * TORSO_HEIGHT,
20,
				20);
		glTranslatef(0.0, 0.0, -0.1 * TORSO_HEIGHT);
		gluCylinder(t, 1.2 * TORSO_RADIUS, TORSO_RADIUS, 0.1 * TORSO_HEIGHT,
20,
				20);
	}

	glPopMatrix();
}

void Head()
{
	glPushMatrix();
	glTranslatef(0.0, 0.9 * TORSO_HEIGHT + HEAD_HEIGHT / 2.0, 0.0);
	Rotate(headRotate);

	// head
	glPushMatrix();
```

```
glColor3f(1.0, 0.9, 0.8);
glScalef(1, 1.1, 1);
gluSphere(h, HEAD_RADIUS, 30, 30);
glPopMatrix();

// eyes
glPushMatrix();
glTranslatef(0.5 * HEAD_RADIUS, 0.3 * HEAD_RADIUS, 0.8 * HEAD_RADIUS);
glColor3f(0.0, 0.0, 0.0);
gluSphere(h, HEAD_RADIUS * 0.2, 10, 10);
glPopMatrix();

glPushMatrix();
glTranslatef(0.0 * HEAD_RADIUS, 0.3 * HEAD_RADIUS, 0.8 * HEAD_RADIUS);
glColor3f(0.0, 0.0, 0.0);
gluSphere(h, HEAD_RADIUS * 0.2, 10, 10);
glPopMatrix();

glPushMatrix();
glTranslatef(-0.5 * HEAD_RADIUS, 0.3 * HEAD_RADIUS, 0.8 * HEAD_RADIUS);
glColor3f(0.0, 0.0, 0.0);
gluSphere(h, HEAD_RADIUS * 0.2, 10, 10);
glPopMatrix();

glPushMatrix();
glTranslatef(0.5 * HEAD_RADIUS, -0.3 * HEAD_RADIUS, 0.8 * HEAD_RADIUS);
glColor3f(0.0, 0.0, 0.0);
gluSphere(h, HEAD_RADIUS * 0.2, 10, 10);
glPopMatrix();

glPushMatrix();
glTranslatef(0.0 * HEAD_RADIUS, -0.3 * HEAD_RADIUS, 0.8 * HEAD_RADIUS);
glColor3f(0.0, 0.0, 0.0);
gluSphere(h, HEAD_RADIUS * 0.2, 10, 10);
glPopMatrix();

glPushMatrix();
glTranslatef(-0.5 * HEAD_RADIUS, -0.3 * HEAD_RADIUS, 0.8 * HEAD_RADIUS);
```

```
	glColor3f(0.0, 0.0, 0.0);
	gluSphere(h, HEAD_RADIUS * 0.2, 10, 10);
	glPopMatrix();

	// hair
	glPushMatrix();
	glColor3f(0.0, 0.0, 0.0);
	glTranslatef(0.0, 0.7 * HEAD_RADIUS, 0.0);
	glRotatef(-90.0, 1.0, 0.0, 0.0);
	gluCylinder(h, 0.5 * HEAD_RADIUS, 1.5 * HEAD_RADIUS, 0.5 * HEAD_RADIUS,
20,
			20);
	glPopMatrix();

	glPopMatrix();
}

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

	// shoulder
	glColor3f(0.8549, 0.64706, 0.12549);
	gluSphere(h, 0.7 * JOINT_RADIUS, 20, 20);

	// arms
	for (int i = 0; i < 2; i++)
	{
		if (i == 0)
		{
			Rotate(rightUpArmRotate);
		}
		else
		{
			Rotate(rightLowArmRotate);
		}
		glTranslatef(0.0, -0.3 * JOINT_RADIUS, 0.0);
```

```
            glPushMatrix();
            glRotatef(90.0, 1.0, 0.0, 0.0);
            glColor3f(1.0, 0.9, 0.9);
            gluCylinder(t, UPPER_ARM_RADIUS, UPPER_ARM_RADIUS,
UPPER_ARM_HEIGHT, 20,
                                20);
            glPopMatrix();

            glTranslatef(0.0, -UPPER_ARM_HEIGHT - 0.3 * JOINT_RADIUS, 0.0);
            gluSphere(h, 0.6 * JOINT_RADIUS, 20, 20);
      }

      glPopMatrix();
}

void LeftArm()
{
      glPushMatrix();
      glTranslatef(-TORSO_RADIUS - 0.1 * JOINT_RADIUS, 0.8 * TORSO_HEIGHT, 0.0);

      // shoulder
      glColor3f(0.8549, 0.64706, 0.12549);
      gluSphere(h, 0.7 * JOINT_RADIUS, 20, 20);

      // arms
      for (int i = 0; i < 2; i++)
      {
            if (i == 0)
            {
                  Rotate(leftUpArmRotate);
            }
            else
            {
                  Rotate(leftLowArmRotate);
            }
            glTranslatef(0.0, -0.3 * JOINT_RADIUS, 0.0);
            glPushMatrix();
            glRotatef(90.0, 1.0, 0.0, 0.0);
```

```
            glColor3f(1.0, 0.9, 0.9);
            gluCylinder(t, UPPER_ARM_RADIUS, UPPER_ARM_RADIUS,
UPPER_ARM_HEIGHT, 20,
                                20);
            glPopMatrix();

            glTranslatef(0.0, -UPPER_ARM_HEIGHT - 0.3 * JOINT_RADIUS, 0.0);
            gluSphere(h, 0.6 * JOINT_RADIUS, 20, 20);
      }

      glPopMatrix();
}

void RightLeg()
{
      glPushMatrix();
      glColor3f(1.0, 0.9, 0.9);

      glTranslatef(0.7 * TORSO_RADIUS, 0.0, 0.0);
      Rotate(rightUpLegRotate);

      gluSphere(h, JOINT_RADIUS, 30, 30);

      glTranslatef(0.0, -0.3 * JOINT_RADIUS, 0.0);
      glPushMatrix();
      glRotatef(90.0, 1.0, 0.0, 0.0);
      gluCylinder(t, 0.6 * UPPER_LEG_RADIUS, 0.6 * UPPER_LEG_RADIUS,
                        UPPER_LEG_HEIGHT, 30, 30);
      glPopMatrix();

      glTranslatef(0.0, -UPPER_LEG_HEIGHT - 0.3 * JOINT_RADIUS, 0.0);
      gluSphere(h, 0.8 * JOINT_RADIUS, 30, 30);

      Rotate(rightLowLegRotate);
      glTranslatef(0.0, -0.3 * JOINT_RADIUS, 0.0);

      glPushMatrix();
      glRotatef(90.0, 1.0, 0.0, 0.0);
```

```
        gluCylinder(t, 1.2 * LOWER_LEG_RADIUS, 1.2 * LOWER_LEG_RADIUS,
                        LOWER_LEG_HEIGHT, 30, 30);
        glPopMatrix();

        glPopMatrix();
}

void LeftLeg()
{
        glPushMatrix();
        glColor3f(1.0, 0.9, 0.9);

        glTranslatef(-0.7 * TORSO_RADIUS, 0.0, 0.0);
        Rotate(leftUpLegRotate);

        gluSphere(h, JOINT_RADIUS, 30, 30);

        glTranslatef(0.0, -0.3 * JOINT_RADIUS, 0.0);
        glPushMatrix();
        glRotatef(90.0, 1.0, 0.0, 0.0);
        gluCylinder(t, 0.6 * UPPER_LEG_RADIUS, 0.6 * UPPER_LEG_RADIUS,
                        UPPER_LEG_HEIGHT, 30, 30);
        glPopMatrix();

        glTranslatef(0.0, -UPPER_LEG_HEIGHT - 0.4 * JOINT_RADIUS, 0.0);
        gluSphere(h, 0.8 * JOINT_RADIUS, 30, 30);

        Rotate(leftLowLegRotate);

        glTranslatef(0.0, -0.3 * JOINT_RADIUS, 0.0);
        glPushMatrix();
        glRotatef(90.0, 1.0, 0.0, 0.0);
        gluCylinder(t, 1.2 * LOWER_LEG_RADIUS, 1.2 * LOWER_LEG_RADIUS,
                        LOWER_LEG_HEIGHT, 30, 30);
        glPopMatrix();

        glPopMatrix();
}
```

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

	glTranslatef(init_Pos[0], init_Pos[1], init_Pos[2]);

	Rotate(init_Rot);

	Rotate(robotRotate);

	Torso();
	Head();
	RightArm();
	LeftArm();
	RightLeg();
	LeftLeg();

	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(int key, int x, int y)
{
	switch (key)
	{
	case GLUT_KEY_UP:
		init_Pos[1] += 0.5;
		glutPostRedisplay();
		break;
	case GLUT_KEY_DOWN:
		init_Pos[1] -= 0.5;
```

```
            glutPostRedisplay();
            break;
        case GLUT_KEY_LEFT:
            init_Pos[0] -= 0.5;
            glutPostRedisplay();
            break;
        case GLUT_KEY_RIGHT:
            init_Pos[0] += 0.5;
            glutPostRedisplay();
            break;
        default:
            break;
        }
}

void resetRotate()
{
        Change(torsoRotate, 0.0, 0.0, 0.0);
        Change(robotRotate, 0.0, 0.0, 0.0);
        Change(headRotate, 0.0, 0.0, 0.0);
        Change(leftUpArmRotate, 0.0, 0.0, -45.0);
        Change(leftLowArmRotate, 0.0, 0.0, 0.0);
        Change(rightUpArmRotate, 0.0, 0.0, 45.0);
        Change(rightLowArmRotate, 0.0, 0.0, 0.0);
        Change(leftUpLegRotate, 0.0, 0.0, 0.0);
        Change(leftLowLegRotate, 0.0, 0.0, 0.0);
        Change(rightUpLegRotate, 0.0, 0.0, 0.0);
        Change(rightLowLegRotate, 0.0, 0.0, 0.0);
}

void menu(int id)
{
        switch (id)
        {
        case 0:
            actionNum = 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;
            resetRotate();
            glutPostRedisplay();
            break;
        case 1:
        case 2:
        case 3:
        case 4:
            resetRotate();
            actionNum = id;
            glutPostRedisplay();
            break;
        case 9:
            exit(0);
            break;
        default:
            break;
        }
}

void cheer(int time)
{
        switch (time % 2)
        {
        case 0:
            Change(leftUpArmRotate, 0.0, 0.0, -180.0);
            Change(rightUpArmRotate, 0.0, 0.0, -180.0);
            break;
        case 1:
            Change(leftUpArmRotate, 0.0, 0.0, 0.0);
            Change(rightUpArmRotate, 0.0, 0.0, 0.0);
            break;
        default:
            break;
        }
```

```
}

void dance(int time)
{
	switch (time % 4)
	{
	case 0:
		Change(leftUpArmRotate, 180.0, 0.0, 0.0);
		Change(leftLowArmRotate, 0.0, 0.0, 0.0);
		Change(rightUpArmRotate, 0.0, 0.0, 90.0);
		Change(rightLowArmRotate, 0.0, 0.0, 0.0);

		Change(leftUpLegRotate, 0.0, 0.0, 0.0);
		Change(leftLowLegRotate, 0.0, 0.0, 0.0);
		Change(rightUpLegRotate, 0.0, 0.0, 60.0);
		Change(rightLowLegRotate, 0.0, 0.0, -40.0);
		break;
	case 2:
		Change(leftUpArmRotate, 0.0, 0.0, -90.0);
		Change(leftLowArmRotate, 0.0, 0.0, 0.0);
		Change(rightUpArmRotate, -180.0, 0.0, 0.0);
		Change(rightLowArmRotate, 0.0, 0.0, 0.0);

		Change(leftUpLegRotate, 0.0, 0.0, -60.0);
		Change(leftLowLegRotate, 0.0, 0.0, 40.0);
		Change(rightUpLegRotate, 0.0, 0.0, 0.0);
		Change(rightLowLegRotate, 0.0, 0.0, 0.0);
		break;
	case 1:
	case 3:
		Change(leftUpArmRotate, 0.0, 0.0, 0.0);
		Change(leftLowArmRotate, 0.0, 0.0, 0.0);
		Change(rightUpArmRotate, 0.0, 0.0, 0.0);
		Change(rightLowArmRotate, 0.0, 0.0, 0.0);

		Change(leftUpLegRotate, 0.0, 0.0, 0.0);
		Change(leftLowLegRotate, 0.0, 0.0, 0.0);
		Change(rightUpLegRotate, 0.0, 0.0, 0.0);
```

```
            Change(rightLowLegRotate, 0.0, 0.0, 0.0);
            break;
        default:
            break;
        }
}

void swim(int time)
{
        Change(robotRotate, 70.0, 0.0, 0.0);
        switch (time % 6)
        {
        case 0:
        case 3:
            Change(leftUpArmRotate, 0.0, 0, 180.0);
            Change(rightUpArmRotate, 0.0, 0.0, 180.0);
            break;
        case 1:
            Change(leftUpArmRotate, 120.0, 0, 180.0);
            Change(rightUpArmRotate, 0.0, 0.0, 180.0);
            break;
        case 2:
            Change(leftUpArmRotate, -120.0, 0, 180.0);
            Change(rightUpArmRotate, 0.0, 0.0, 180.0);
            break;
        case 4:
            Change(leftUpArmRotate, 0.0, 0, 180.0);
            Change(rightUpArmRotate, 120.0, 0.0, 180.0);
            break;
        case 5:
            Change(leftUpArmRotate, 0.0, 0, 180.0);
            Change(rightUpArmRotate, -120.0, 0.0, 180.0);
            break;

        default:
            break;
        }
}
```

```
void kick(int time)
{
	switch (time % 3)
	{
	case 0:
		Change(leftUpArmRotate, 0.0, 0.0, -40.0);
		Change(leftLowArmRotate, -65.0, 0.0, 40.0);
		Change(rightUpArmRotate, 0.0, 0.0, 40.0);
		Change(rightLowArmRotate, -65.0, 0.0, -20.0);

		Change(leftUpLegRotate, 0.0, 0.0, 0.0);
		Change(leftLowLegRotate, 10.0, 0.0, 0.0);
		Change(rightUpLegRotate, 0.0, 0.0, 0.0);
		Change(rightLowLegRotate, 10.0, 0.0, 0.0);
		break;
	case 1:
		Change(leftUpArmRotate, 120.0, 0.0, -40.0);
		Change(leftLowArmRotate, -65.0, 0.0, 40.0);
		Change(rightUpArmRotate, 0.0, 0.0, 40.0);
		Change(rightLowArmRotate, -65.0, 0.0, -20.0);

		Change(leftUpLegRotate, 0.0, 0.0, 0.0);
		Change(leftLowLegRotate, 90.0, 0.0, 0.0);
		Change(rightUpLegRotate, 0.0, 0.0, 0.0);
		Change(rightLowLegRotate, 10.0, 0.0, 0.0);
		break;
	case 2:
		Change(leftUpArmRotate, 0.0, 0.0, -40.0);
		Change(leftLowArmRotate, -65.0, 0.0, 40.0);
		Change(rightUpArmRotate, 0.0, 0.0, 40.0);
		Change(rightLowArmRotate, -65.0, 0.0, -20.0);

		Change(leftUpLegRotate, 0.0, 0.0, 0.0);
		Change(leftLowLegRotate, -60.0, 0.0, 0.0);
		Change(rightUpLegRotate, 0.0, 0.0, 0.0);
		Change(rightLowLegRotate, 10.0, 0.0, 0.0);
		break;
```

```
        default:
                break;
        }
}

void action()
{
        long time_box;
        time_box = time(0);
        switch (actionNum)
        {
        case 1:
                cheer(time_box);
                break;
        case 2:
                swim(time_box);
                break;
        case 3:
                dance(time_box);
                break;
        case 4:
                kick(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(500, 700);
        glutCreateWindow("Robot");
        init();
```

```
    glutCreateMenu(menu);
    glutAddMenuEntry("reset", 0);
    glutAddMenuEntry("cheer", 1);
    glutAddMenuEntry("swim", 2);
    glutAddMenuEntry("dance", 3);
    glutAddMenuEntry("kick", 4);
    glutAddMenuEntry("quit", 9);
    glutAttachMenu(GLUT_RIGHT_BUTTON);

    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutMouseFunc(mouseButton);
    glutMotionFunc(mouseMotion);
    glutSpecialFunc(keyboard);
    glutIdleFunc(action);
    glutMainLoop();

    return 0;
}
```