# 《机器人可视化》

题目：机器人及其各类动作可视化

目录

[《机器人可视化》 1](#_Toc165283859)

[1. 项目需求简介 3](#_Toc165283860)

[2. 项目详细设计及成果展示 3](#_Toc165283861)

[绘制窗口 3](#_Toc165283862)

[Menu菜单 4](#_Toc165283863)

[机器人组件 5](#_Toc165283864)

[各组件的参数 6](#_Toc165283865)

[机器人运动 7](#_Toc165283866)

[**打招呼** 7](#_Toc165283867)

[**摊手** 9](#_Toc165283868)

[**原地踏步** 9](#_Toc165283869)

[**马步下蹲** 12](#_Toc165283870)

[**自旋** 15](#_Toc165283871)

[附录 16](#_Toc165283872)

[Menu代码： 16](#_Toc165283873)

[LYQROB源文件代码： 17](#_Toc165283874)

# 项目需求简介

随着新时代机器人技术的快速发展，机器人在我们的日常社会中的重要性日益凸显。机器人在工业生产中减少了人类在危险环境下的风险，执行重复、繁琐的任务释放了人类的时间和精力，提高了生产效率和产品质量；在医疗领域，机器人手术系统和辅助设备已经成为医生的得力助手，提高了手术精度和治疗效果；在服务行业，机器人可以24/7接待客人、提供信息咨询、办理登记手续，它们可以不间断地提供服务，而且在多语言环境中表现出色。因此，发展机器人的重要性不言而喻。

那么要设计一款好的机器人，首先就需要在软件当中设计出机器人的外形以及动作功能。那么在计算机中，使用C/C++语言下的OpenGL库就很好的满足了这一需求。

OpenGL（Open Graphics Library）是一种跨平台的图形渲染API（应用程序编程接口），由Khronos Group维护和更新。它提供了一套标准的函数接口，用于进行2D和3D图形渲染，以及其他与图形相关的操作。

OpenGL具有跨平台性、灵活性、硬件加速、开放标准、工具丰富等特点和功能。通过使用OpenGL库，计算机就可以很好的把三维世界中的三维物体（例如本次项目需求中的机器人）在二维屏幕上显示出来。

# 项目详细设计及成果展示

## 绘制窗口

为了展示机器人的外形以及运动，首先需要创建一个窗口。核心代码如下：

*int main(int argc, char\*\* argv)*

*{*

*glutInit(&argc, argv);*

*glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);*

*glutInitWindowSize(500, 500);*

*glutInitWindowPosition(100, 100);*

*glutCreateWindow("刘英琦课程设计大作业");*

*glutCreateMenu(OnMenu);*

*glutDisplayFunc(onDisplay);*

*glutTimerFunc(0, LYQ::update, 0); // 开始更新动画*

*makeMenu();*

*glEnable(GL\_DEPTH\_TEST);*

*glutMainLoop();*

*return 0;*

*}*

## Menu菜单

在窗口中，我设计了一个Menu用以选择机器人进行不同的运动，核心代码见附录。

接下来，开始设计本次的机器人。为了有助于模块化编程模块化编程：将函数的声明与实现分离。这样做使得代码结构更加清晰，方便维护和管理。将机器人的设计细节都封装到LYQROB文件中，LYQROB头文件如下：

*#ifndef LYQROB*

*#define LYQROB*

*#include "stdafx.h"*

*#include <glut.h>*

*#include "global\_variables.h"*

*#include "init\_gv.h"*

*#include <Windows.h>*

*#include <math.h>*

*class LYQ*

*{*

*public:*

*LYQ();*

*~LYQ();*

*static int flag;*

*static void init();*

*static void drawRobot();*

*static void onDisplay();*

*static void update(int value);*

*};*

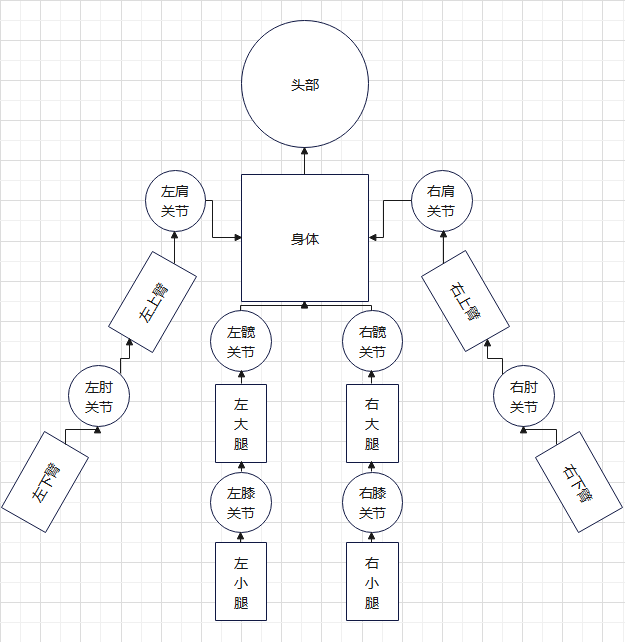
*#endif*

## 机器人组件

在LYQROB中，首先画出机器人的整体结构，包括机器人的头部，身体，左右肩关节、左右肘关节、左右髋关节、左右膝关节八个核心关节，左右大臂、左右小臂、左右大腿、左右小腿八个核心肢体，以及帽子，眼睛，嘴巴，领结等装饰品。这些共同组成了机器人的组件。如下图所示，核心代码见附录：



在创建组件时，我通过特定的矩阵的压栈出栈操作，使得核心组件呈现了一定的层次关系，如图所示：



## 各组件的参数

有了这些组件后，我通过在头文件中申明、在源文件中定义，设定了一系列全局变量参数，并封装成global\_variables文件，用以细致的操控这些组件，并在LYQROB文件中引用该文件，以便于协调的完成动作，头文件代码如下：

*#ifndef GLOBAL\_VARIABLES\_H*

*#define GLOBAL\_VARIABLES\_H*

*extern double pi;*

*extern float body\_center\_x;*

*extern float body\_center\_y;*

*extern float body\_center\_z;*

*extern float left\_upper\_arm\_swing\_angle;*

*extern float right\_upper\_arm\_swing\_angle;*

*extern float left\_lower\_arm\_swing\_angle;*

*extern float right\_lower\_arm\_swing\_angle;*

*extern float left\_upper\_leg\_swing\_angle;*

*extern float right\_upper\_leg\_swing\_angle;*

*extern float left\_lower\_leg\_swing\_angle;*

*extern float right\_lower\_leg\_swing\_angle;*

*extern float LeftUpperArmSpeed;*

*extern float LeftUpperArmRange;*

*extern float RightUpperArmSpeed;*

*extern float RightUpperArmRange;*

*extern float LeftLowerArmSpeed;*

*extern float LeftLowerArmRange;*

*extern float RightLowerArmSpeed;*

*extern float RightLowerArmRange;*

*extern float LeftUpperLegSpeed;*

*extern float LeftUpperLegRange;*

*extern float RightUpperLegSpeed;*

*extern float RightUpperLegRange;*

*extern float LeftLowerLegSpeed;*

*extern float LeftLowerLegRange;*

*extern float RightLowerLegSpeed;*

*extern float RightLowerLegRange;*

*extern float sp;*

*#endif*

有了组件和参数后，接下来我由易到难设计了一系列的机器人的基础动作，并将其可视化。

## 机器人运动

### **打招呼**

在设计机器人打招呼的运动状态，只需要考虑一只手臂的运动关系，我们以左臂为例，需要考虑左上臂绕左肩关节旋转和左下臂绕左肘关节旋转。我将其封装成GREET.h，届时供LYQROB直接调用，代码和运行截图如下：

*#ifndef GREET*

*#define GREET*

*#include "global\_variables.h"*

*void Greet(float currentTime, float& body\_center\_x, float& body\_center\_y, float& body\_center\_z,float& left\_upper\_arm\_swing\_angle, float& right\_upper\_arm\_swing\_angle,float& left\_lower\_arm\_swing\_angle, float& right\_lower\_arm\_swing\_angle,float& left\_upper\_leg\_swing\_angle, float& right\_upper\_leg\_swing\_angle,float& left\_lower\_leg\_swing\_angle, float& right\_lower\_leg\_swing\_angle){*

*LeftUpperArmSpeed = 0.7;*

*LeftUpperArmRange = 72;*

*LeftLowerArmSpeed = 4.2;*

*LeftLowerArmRange = 28;*

*left\_upper\_arm\_swing\_angle = sin(currentTime \* LeftUpperArmSpeed) \* LeftUpperArmRange;*

*if (left\_upper\_arm\_swing\_angle > 0)*

*{*

*left\_upper\_arm\_swing\_angle = -left\_upper\_arm\_swing\_angle;*

*}*

*if (left\_upper\_arm\_swing\_angle < -36)*

*{*

*left\_lower\_arm\_swing\_angle = sin(currentTime \* LeftLowerArmSpeed) \* LeftLowerArmRange;*

*if (left\_lower\_arm\_swing\_angle > 0)*

*{*

*left\_lower\_arm\_swing\_angle = -left\_lower\_arm\_swing\_angle;*

*}*

*}*

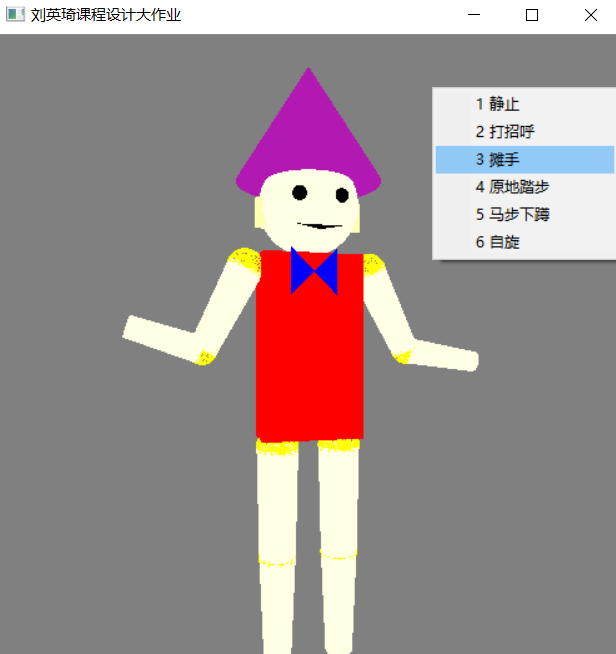
*}*

*#endif //!GREET*



### **摊手**

比起打招呼，摊手则需要考虑两只上臂不同的旋转速率比例，以及两只小臂基于对应大臂的旋转速度和旋转范围，相比起打招呼自然复杂一些。我将其封装成SPREADHANDS.h，届时供LYQROB直接调用，运行截图如下：



### **原地踏步**

原地踏步不仅需要考虑两只手臂的摆动速率，还需要考虑两条腿的协调性。我将其封装成MARCHONTHESPOT.h，届时供LYQROB直接调用，代码和运行截图如下：

*#ifndef MARCHONTHESPOT*

*#define MARCHONTHESPOT*

*#include "global\_variables.h"*

*void MarchOnTheSpot(float currentTime, float& body\_center\_x, float& body\_center\_y, float& body\_center\_z,float& left\_upper\_arm\_swing\_angle, float& right\_upper\_arm\_swing\_angle,float& left\_lower\_arm\_swing\_angle, float& right\_lower\_arm\_swing\_angle,float& left\_upper\_leg\_swing\_angle, float& right\_upper\_leg\_swing\_angle,float& left\_lower\_leg\_swing\_angle, float& right\_lower\_leg\_swing\_angle) {*

*LeftUpperLegRange = RightUpperLegRange = LeftLowerLegRange = RightLowerLegRange = 60;*

*LeftLowerArmRange = RightLowerArmRange = 45;*

*// 计算左右臂摆动角度*

*left\_upper\_arm\_swing\_angle = (sin(currentTime \* LeftUpperArmSpeed) + cos(currentTime \* LeftUpperArmSpeed)) / 2 \* LeftUpperArmRange*

*right\_upper\_arm\_swing\_angle = (sin(currentTime \* RightUpperArmSpeed) + cos(currentTime \* RightUpperArmSpeed)) / 2 \* RightUpperArmRange;*

*// 计算左右前臂摆动角度*

*left\_lower\_arm\_swing\_angle = (sin(currentTime \* LeftLowerArmSpeed) + cos(currentTime \* LeftLowerArmSpeed)) / 2 \* LeftLowerArmRange;*

*if (left\_lower\_arm\_swing\_angle > -3)*

*{*

*left\_lower\_arm\_swing\_angle = 0;*

*}*

*right\_lower\_arm\_swing\_angle = (sin(currentTime \* RightLowerArmSpeed) + cos(currentTime \* RightLowerArmSpeed)) / 2 \* RightLowerArmRange;*

*if (right\_lower\_arm\_swing\_angle < 3)*

*{*

*right\_lower\_arm\_swing\_angle = 0;*

*}*

*// 计算左右大腿摆动角度*

*left\_upper\_leg\_swing\_angle = (sin(currentTime \* LeftUpperLegSpeed) + cos(currentTime \* LeftUpperLegSpeed)) / 2 \* LeftUpperLegRange;*

*if (left\_upper\_leg\_swing\_angle < 0)*

*{*

*left\_upper\_leg\_swing\_angle = 0;*

*}*

*right\_upper\_leg\_swing\_angle = (sin(currentTime \* RightUpperLegSpeed) + cos(currentTime \* RightUpperLegSpeed)) / 2 \* RightUpperLegRange;*

*if (right\_upper\_leg\_swing\_angle > 0)*

*{*

*right\_upper\_leg\_swing\_angle = 0;*

*}*

*// 计算左右小腿摆动角度*

*left\_lower\_leg\_swing\_angle = (sin(currentTime \* LeftLowerLegSpeed) + cos(currentTime \* LeftLowerLegSpeed)) / 2 \* LeftLowerLegRange;*

*if (left\_lower\_leg\_swing\_angle < 0)*

*{*

*left\_lower\_leg\_swing\_angle = 0;*

*}*

*right\_lower\_leg\_swing\_angle = (sin(currentTime \* RightLowerLegSpeed) + cos(currentTime \* RightLowerLegSpeed)) / 2 \* RightLowerLegRange;*

*if (right\_lower\_leg\_swing\_angle > 0)*

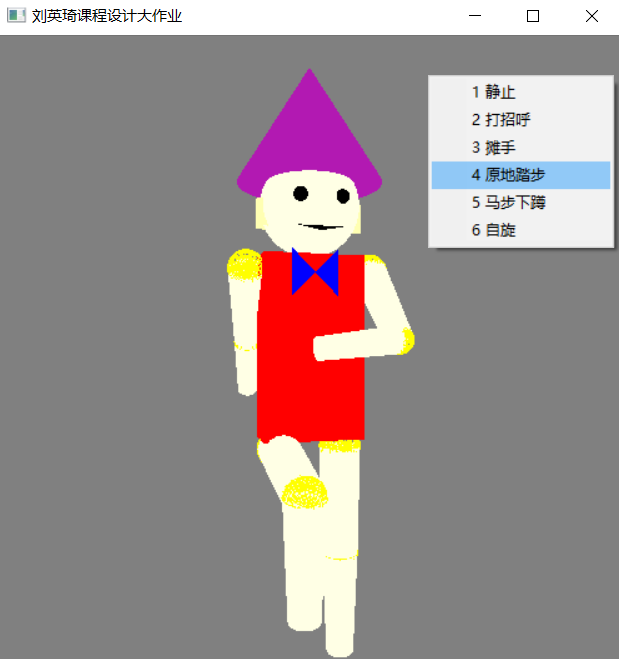
*{*

*right\_lower\_leg\_swing\_angle = 0;*

*}*

*}*

*#endif // !MARCHONTHESPOT*



### **马步下蹲**

与原地踏步类似，但是马步下蹲动作除了考虑各肢体旋转的一致性，还需要考虑当腿部旋转式，机器人整体坐标的移动。通过机器人的整体结构可知，只需要移动头部的坐标即可达到所需要的效果.我将这个逻辑写成了代码，并将其封装在了HOUSERIDINGSTANCE.h之中，届时供LYQROB直接调用，核心代码及效果图如下：

*#ifndef HORSERIDINGSTANCE*

*#define HORSERIDINGSTANCE*

*#include "global\_variables.h"*

*void HorseRidingStance(float currentTime, float& body\_center\_x, float& body\_center\_y, float& body\_center\_z,float& left\_upper\_arm\_swing\_angle, float& right\_upper\_arm\_swing\_angle,float& left\_lower\_arm\_swing\_angle, float& right\_lower\_arm\_swing\_angle,float& left\_upper\_leg\_swing\_angle, float& right\_upper\_leg\_swing\_angle,float& left\_lower\_leg\_swing\_angle, float& right\_lower\_leg\_swing\_angle){*

*float speed = 1;*

*LeftUpperLegSpeed = RightUpperLegSpeed = LeftLowerLegSpeed = RightLowerLegSpeed = speed / 2 ;*

*LeftUpperArmSpeed = RightUpperArmSpeed = LeftLowerArmSpeed = RightLowerArmSpeed = speed / 2;*

*LeftUpperLegRange = 45;*

*RightUpperLegRange = 45;*

*LeftLowerLegRange = 45;*

*RightLowerLegRange = 45;*

*body\_center\_y = (cos(currentTime\*speed) - 1)\* 0.5;*

*body\_center\_z = -fabs( 0.99 \* sin(currentTime\*speed / 2));*

*//printf("%f\n",body\_center\_z);*

*left\_upper\_leg\_swing\_angle = fabs(sin(currentTime \* LeftUpperLegSpeed) \* LeftUpperLegRange);*

*right\_upper\_leg\_swing\_angle = fabs(sin(currentTime \* RightUpperLegSpeed) \* RightUpperLegRange);*

*left\_lower\_leg\_swing\_angle = fabs(sin(currentTime \* LeftLowerLegSpeed) \* LeftLowerLegRange);*

*right\_lower\_leg\_swing\_angle = fabs(sin(currentTime \* RightLowerLegSpeed) \* RightLowerLegRange);*

*left\_upper\_arm\_swing\_angle = fabs(sin(currentTime \* LeftUpperArmSpeed)\* LeftUpperArmRange);*

*right\_upper\_arm\_swing\_angle = fabs(sin(currentTime \* RightUpperArmSpeed)\* RightUpperArmRange);*

*left\_lower\_arm\_swing\_angle = fabs(sin(currentTime \* LeftLowerArmSpeed) \* LeftLowerArmRange);*

*right\_lower\_arm\_swing\_angle = -fabs(sin(currentTime \* RightLowerArmSpeed) \* RightLowerArmRange);*

*}*

*#endif // !HORSERIDINGSTANCE*



### **自旋**

自旋相比其其他动作略有不同，是在设计完其他动作之后，想要加入的一个旋转效果，使得机器人的各种动作类型均被设计过。有关旋转，使用的核心代码是：

glRotatef(sp, 0, 1, 0)

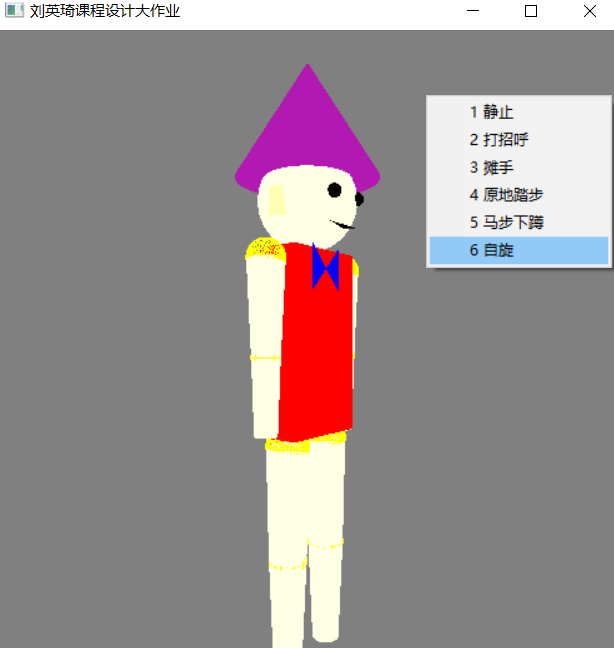
通过调节sp的数值，然后在display函数中，加入条件语句：

if (flag == 5){

glutIdleFunc(spin);

}

达到机器人的旋转效果，运行截图如下图所示：



# 附录

## Menu代码：

*#define ROBSTATIC 1 //静态*

*#define GRT 2 //打招呼*

*#define SH 3 //摊手*

*#define MOTS 4 //原地踏步*

*#define HRS 5 //马步下蹲*

*#define SPIN 6 //自旋*

*GLint nSelected = ROBSTATIC;*

*void makeMenu()*

*{*

*glutAddMenuEntry("1 静止", ROBSTATIC);*

*glutAddMenuEntry("2 打招呼", GRT);*

*glutAddMenuEntry("3 摊手", SH);*

*glutAddMenuEntry("4 原地踏步", MOTS);*

*glutAddMenuEntry("5 马步下蹲", HRS);*

*glutAddMenuEntry("6 自旋", SPIN);*

*glutAttachMenu(GLUT\_RIGHT\_BUTTON); //指定菜单事件由鼠标右键单击产生*

*}*

*void OnMenu(int value)*

*{*

*switch (value)*

*{*

*case ROBSTATIC:*

*nSelected = ROBSTATIC;*

*break;*

*case GRT:*

*nSelected = GRT;*

*break;*

*case SH:*

*nSelected = SH;*

*break;*

*case MOTS:*

*nSelected = MOTS;*

*break;*

*case HRS:*

*nSelected = HRS;*

*break;*

*case SPIN:*

*nSelected = SPIN;*

*break;*

*default:*

*break;*

*}*

*glutPostRedisplay();*

*}*

*void onDisplay(){*

*LYQ::onDisplay();*

*switch (nSelected)*

*{*

*case ROBSTATIC:*

*LYQ::flag = 0;*

*break;*

*case GRT:*

*LYQ::flag = 1;*

*break;*

*case SH:*

*LYQ::flag = 2;*

*break;*

*case MOTS:*

*LYQ::flag = 3;*

*break;*

*case HRS:*

*LYQ::flag = 4;*

*break;*

*case SPIN:*

*LYQ::flag = 5;*

*break;*

*default:*

*break;*

*}*

*}*

## LYQROB源文件代码：

*#include "stdafx.h"*

*#include"LYQROB.h"*

*#include "GREET.h"*

*#include "SPREADHANDS.h"*

*#include"MARCHONTHESPOT.h"*

*#include "HORSERIDINGSTANCE.h"*

*int LYQ::flag = 0;*

*void drawhat(float radius, float height, int segments) {*

*// 绘制侧面*

*glColor3f(0.7, 0.1, 0.7);*

*glBegin(GL\_TRIANGLES);*

*for (int i = 0; i < segments; ++i) {*

*float angle1 = 2.0f \* pi \* (float)i / (float)segments;*

*float angle2 = 2.0f \* pi \* (float)(i + 1) / (float)segments;*

*float x1 = radius \* cos(angle1);*

*float z1 = radius \* sin(angle1);*

*float x2 = radius \* cos(angle2);*

*float z2 = radius \* sin(angle2);*

*// 侧面三角形*

*glVertex3f(x1, 0, z1); // 底面三角形顶点1*

*glVertex3f(x2, 0, z2); // 底面三角形顶点2*

*glVertex3f(0, height, 0); // 顶部顶点*

*}*

*glEnd();*

*// 绘制底面*

*glBegin(GL\_TRIANGLE\_FAN);*

*glVertex3f(0, 0, 0); // 圆锥底部中心*

*for (int i = 0; i <= segments; ++i) {*

*float angle = 2.0f \* pi \* (float)i / (float)segments;*

*float x = radius \* cos(angle);*

*float z = radius \* sin(angle);*

*glVertex3f(x, 0, z); // 底面顶点*

*}*

*glEnd();*

*}*

*void bowtie() {*

*glRotatef(90.0, 0.0, 0.0, 1.0); // 绕着 z 轴旋转 90 度*

*glBegin(GL\_TRIANGLES);*

*glColor3f(0.0, 0.0, 1.0);*

*glVertex2f(-0.25, 0.25); // 左上角顶点*

*glVertex2f(0.25, 0.25); // 右上角顶点*

*glVertex2f(0.0, 0.0); // 中心顶点*

*glEnd();*

*glBegin(GL\_TRIANGLES);*

*glVertex2f(-0.25, -0.25); // 左下角顶点*

*glVertex2f(0.25, -0.25); // 右下角顶点*

*glVertex2f(0.0, 0.0); // 中心顶点*

*glEnd();*

*}*

*void mouse(){*

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

*glBegin(GL\_TRIANGLES);*

*glVertex3f(0.0, 0.0, -0.25);*

*glVertex3f(-0.01, -0.05, 0.0);*

*glVertex3f(0.0, 0.0, 0.25);*

*glEnd();*

*}*

*void LYQ::init() {*

*glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);*

*glMatrixMode(GL\_PROJECTION);*

*glLoadIdentity();*

*gluPerspective(60, 1.0, 1.0, 100.0);*

*glMatrixMode(GL\_MODELVIEW);*

*glLoadIdentity();*

*}*

*void color\_skin()*

*{*

*float red = 1.0;*

*float green = 1.0;*

*float blue = 0.9;*

*glColor3f(red, green, blue);*

*}*

*void color\_joint()*

*{*

*float red = 1.0;*

*float green = 1.0;*

*float blue = 0.0;*

*glColor3f(red, green, blue);*

*}*

*void LYQ::drawRobot() {*

*glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);*

*if (flag == 5){//自旋操作*

*body\_center\_x = 0;*

*body\_center\_y = 0;*

*body\_center\_z = 0;*

*glRotatef(sp, 0, 1, 0);*

*}*

*// 画头部*

*glPushMatrix();*

*color\_skin(); // 设置头部颜色*

*glTranslatef(body\_center\_x, body\_center\_y, body\_center\_z);*

*glTranslatef(0.0, 1.55, 0.0);*

*glutSolidSphere(0.55, 30, 30);*

*//画耳朵*

*glPushMatrix();*

*glColor3f(1.0, 1.0, 0.7);*

*glRotatef(90.0, 1.0, 0.0, 0.0); // 将圆柱体旋转使其垂直于xy平面*

*glTranslatef(0.5, 0.0, -0.05);*

*GLUquadricObj \*leftear = gluNewQuadric(); // 创建圆柱体对象*

*gluCylinder(leftear, 0.1, 0.1, 0.3, 20, 20); // 绘制圆柱体*

*glPopMatrix();*

*glPushMatrix();*

*glColor3f(1.0, 1.0, 0.7);*

*glRotatef(90.0, 1.0, 0.0, 0.0); // 将圆柱体旋转使其垂直于xy平面*

*glTranslatef(-0.5, 0.0, -0.05);*

*GLUquadricObj \*rightear = gluNewQuadric(); // 创建圆柱体对象*

*gluCylinder(rightear, 0.1, 0.1, 0.3, 20, 20); // 绘制圆柱体*

*glPopMatrix();*

*//画眼睛*

*glPushMatrix();*

*glColor3f(0.0, 0.0, 0.0); // 设置眼睛颜色*

*glTranslatef(0.22, 0.0, 0.5);*

*glutSolidSphere(0.075, 30, 30);*

*glPopMatrix();*

*glPushMatrix();*

*glColor3f(0.0, 0.0, 0.0);*

*glTranslatef(-0.22, 0.0, 0.5);*

*glutSolidSphere(0.075, 30, 30);*

*glPopMatrix();*

*//画嘴巴*

*glPushMatrix();*

*glTranslatef(0.0, -0.3, 0.46);*

*mouse();*

*glPopMatrix();*

*//画帽子*

*glPushMatrix();*

*glTranslatef(0.0, 0.25, 0.0);*

*drawhat(0.8, 1.3, 50);*

*glPopMatrix();*

*// 画身体*

*glPushMatrix();*

*glColor3f(1.0, 0.0, 0.0); // 设置身体颜色*

*glTranslatef(0.0, -1.55, 0.0);*

*glScalef(1.1, 2.0, 0.5);*

*glutSolidCube(1.0);*

*glScalef(1 / 1.1, 1 / 2.0, 1 / 0.5);*

*//画领结*

*glPushMatrix();*

*glTranslatef(0.0, 0.8, 0.252);*

*bowtie();*

*glPopMatrix();*

*// 左肩关节*

*glPushMatrix();*

*color\_joint();*

*glTranslatef(-0.7, 0.85, 0.0);*

*glutWireSphere(0.2, 20, 20);*

*// 左上臂*

*glPushMatrix();*

*color\_skin();*

*// 应用摆动角度*

*switch (flag)*

*{*

*case 0:*

*//静止*

*break;*

*case 1:*

*//打招呼*

*glRotatef(2 \* left\_upper\_arm\_swing\_angle, 0.0, 0.0, 1.0);*

*break;*

*case 2:*

*//摊手*

*glRotatef(left\_upper\_arm\_swing\_angle, 0.0, 0.0, 1.0);*

*break;*

*case 3:*

*//原地踏步*

*glRotatef(2 \* left\_upper\_arm\_swing\_angle, 1.0, 0.0, 0.15);*

*break;*

*case 4:*

*//马步下蹲*

*glRotatef(0.7\* left\_upper\_arm\_swing\_angle, 1.0, -1.0, 0.0);*

*break;*

*default:*

*break;*

*}*

*glRotatef(90.0, 1.0, 0.0, 0.0); // 将圆柱体旋转使其垂直于xy平面*

*GLUquadricObj \*leftUpperArmCylinder = gluNewQuadric(); // 创建圆柱体对象*

*gluCylinder(leftUpperArmCylinder, 0.2, 0.15, 1.0, 20, 20); // 绘制圆柱体*

*// 左肘关节*

*glPushMatrix();*

*color\_joint();*

*glTranslatef(0.0, 0.0, 1.0);*

*glutWireSphere(0.15, 20, 20);*

*// 左下臂*

*glPushMatrix();*

*color\_skin();*

*// 应用摆动角度*

*switch (flag)*

*{*

*case 0:*

*//静止*

*break;*

*case 1:*

*//打招呼*

*glRotatef(3 \* left\_lower\_arm\_swing\_angle, 0.0, 1.0, 0.0);*

*break;*

*case 2:*

*//摊手*

*glRotatef(3 \* left\_lower\_arm\_swing\_angle, 0.0, 1.0, 0.0);*

*break;*

*case 3:*

*//原地踏步*

*glRotatef(3 \* left\_lower\_arm\_swing\_angle, 0.0, -1.0, 0.0);*

*break;*

*case 4:*

*//马步下蹲*

*glRotatef(3.7 \* left\_lower\_arm\_swing\_angle, -1.0, 0.18, 0.0);*

*break;*

*default:*

*break;*

*}*

*GLUquadricObj \*leftLowerArmCylinder = gluNewQuadric();*

*gluCylinder(leftLowerArmCylinder, 0.15, 0.12, 0.8, 20, 20);*

*glPopMatrix();*

*glPopMatrix();*

*glPopMatrix();*

*glPopMatrix();*

*// 右肩关节*

*glPushMatrix();*

*color\_joint();*

*glTranslatef(0.7, 0.85, 0.0);*

*glutWireSphere(0.2, 20, 20);*

*// 右上臂*

*glPushMatrix();*

*color\_skin();*

*// 应用摆动角度*

*switch (flag)*

*{*

*case 0:*

*//静止*

*break;*

*case 1:*

*//打招呼*

*break;*

*case 2:*

*//摊手*

*glRotatef(right\_upper\_arm\_swing\_angle, 0.0, 0.0, -1.0);*

*break;*

*case 3:*

*//原地踏步*

*glRotatef(2 \* right\_upper\_arm\_swing\_angle, -1.0, 0.0, 0.15);*

*break;*

*case 4:*

*//马步下蹲*

*glRotatef(0.7 \* right\_upper\_arm\_swing\_angle, 1.0, -1.0, 0.0);*

*break;*

*default:*

*break;*

*}*

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

*GLUquadricObj \*rightUpperArmCylinder = gluNewQuadric();*

*gluCylinder(rightUpperArmCylinder, 0.2, 0.15, 1.0, 20, 20);*

*// 右肘关节*

*glPushMatrix();*

*color\_joint();*

*glTranslatef(0.0, 0.0, 1.0);*

*glutWireSphere(0.15, 20, 20);*

*// 右下臂*

*glPushMatrix();*

*color\_skin();*

*// 应用摆动角度*

*switch (flag)*

*{*

*case 0:*

*//静止*

*break;*

*case 1:*

*//打招呼*

*break;*

*case 2:*

*//摊手*

*glRotatef(2 \* right\_lower\_arm\_swing\_angle, 0.0, -1.0, 0.0);*

*break;*

*case 3:*

*//原地踏步*

*glRotatef(3 \* right\_lower\_arm\_swing\_angle, 0.0, -1.0, 0.0);*

*break;*

*case 4:*

*//马步下蹲*

*glRotatef(3.7 \* right\_lower\_arm\_swing\_angle, 1.0, -0.18, 0.0);*

*break;*

*default:*

*break;*

*}*

*GLUquadricObj \*rightLowerArmCylinder = gluNewQuadric();*

*gluCylinder(rightLowerArmCylinder, 0.15, 0.12, 0.9, 20, 20);*

*glPopMatrix();*

*glPopMatrix();*

*glPopMatrix();*

*glPopMatrix();*

*// 左髋关节*

*glPushMatrix();*

*color\_joint();*

*glTranslatef(-0.35, -1.1, 0.0);*

*glutWireSphere(0.23, 20, 20);*

*// 左大腿*

*glPushMatrix();*

*color\_skin();*

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

*// 应用摆动角度*

*switch (flag)*

*{*

*case 0:*

*//静止*

*break;*

*case 1:*

*//打招呼*

*break;*

*case 2:*

*//摊手*

*break;*

*case 3:*

*//原地踏步*

*glRotatef(2 \* left\_upper\_leg\_swing\_angle, -1.0, 0.0, 0.0);*

*break;*

*case 4:*

*//马步下蹲*

*glRotatef(2 \* left\_upper\_leg\_swing\_angle, -1.0, -0.4, 0.0);*

*break;*

*default:*

*break;*

*}*

*GLUquadricObj \*leftUpperLegCylinder = gluNewQuadric();*

*gluCylinder(leftUpperLegCylinder, 0.23, 0.2, 1.2, 20, 20);*

*// 左膝关节*

*glPushMatrix();*

*color\_joint();*

*glTranslatef(0.0, 0.0, 1.2);*

*glutWireSphere(0.2, 20, 20);*

*// 左小腿*

*glPushMatrix();*

*color\_skin();*

*// 应用摆动角度*

*switch (flag)*

*{*

*case 0:*

*//静止*

*break;*

*case 1:*

*//打招呼*

*break;*

*case 2:*

*//摊手*

*break;*

*case 3:*

*//原地踏步*

*glRotatef(2 \* left\_lower\_leg\_swing\_angle, 1.0, 0.0, 0.0);*

*break;*

*case 4:*

*//马步下蹲*

*glRotatef(2 \* left\_lower\_leg\_swing\_angle, 1.0, 0.4, 0.0);*

*break;*

*default:*

*break;*

*}*

*GLUquadricObj \*leftLowerLegCylinder = gluNewQuadric();*

*gluCylinder(leftLowerLegCylinder, 0.2, 0.15, 1.1, 20, 20);*

*glPopMatrix();*

*glPopMatrix();*

*glPopMatrix();*

*glPopMatrix();*

*// 右髋关节*

*glPushMatrix();*

*color\_joint();*

*glTranslatef(0.35, -1.1, 0.0);*

*glutWireSphere(0.23, 20, 20);*

*// 右大腿*

*glPushMatrix();*

*color\_skin();*

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

*// 应用摆动角度*

*switch (flag)*

*{*

*case 0:*

*//静止*

*break;*

*case 1:*

*//打招呼*

*break;*

*case 2:*

*//摊手*

*break;*

*case 3:*

*//原地踏步*

*glRotatef(2 \* right\_upper\_leg\_swing\_angle, 1.0, 0.0, 0.0);*

*break;*

*case 4:*

*//马步下蹲*

*glRotatef(2 \* right\_upper\_leg\_swing\_angle, -1.0, 0.4, 0.0);*

*break;*

*default:*

*break;*

*}*

*GLUquadricObj \*rightUpperLegCylinder = gluNewQuadric();*

*gluCylinder(rightUpperLegCylinder, 0.23, 0.2, 1.2, 20, 20);*

*// 右膝关节*

*glPushMatrix();*

*color\_joint();*

*glTranslatef(0.0, 0.0, 1.2);*

*glutWireSphere(0.2, 20, 20);*

*// 右小腿*

*glPushMatrix();*

*color\_skin();*

*// 应用摆动角度*

*switch (flag)*

*{*

*case 0:*

*//静止*

*break;*

*case 1:*

*//打招呼*

*break;*

*case 2:*

*//摊手*

*break;*

*case 3:*

*//原地踏步*

*glRotatef(2 \* right\_lower\_leg\_swing\_angle, -1.0, 0.0, 0.0);*

*break;*

*case 4:*

*//马步下蹲*

*glRotatef(2 \* right\_lower\_leg\_swing\_angle, 1.0, -0.4, 0.0);*

*break;*

*default:*

*break;*

*}*

*GLUquadricObj \*rightLowerLegCylinder = gluNewQuadric();*

*gluCylinder(rightLowerLegCylinder, 0.2, 0.15, 1.1, 20, 20);*

*glPopMatrix();*

*glPopMatrix();*

*glPopMatrix();*

*glPopMatrix();*

*glPopMatrix();*

*glPopMatrix();*

*glFlush();*

*}*

*void spin()*

*{*

*sp += 0.1;*

*glutPostRedisplay();*

*}*

*void LYQ::onDisplay()*

*{*

*// 设置窗口背景颜色*

*glClearColor(0.5f, 0.5f, 0.5f, 1.0f);*

*init();*

*gluLookAt(-1.5, 0, 5.8, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);*

*drawRobot();*

*if (flag == 5){*

*glutIdleFunc(spin);*

*}*

*}*

*void LYQ::update(int value)*

*{*

*float currentTime = glutGet(GLUT\_ELAPSED\_TIME) / 1000.0; // 获取自程序运行开始以来经过的时间（秒）*

*switch (flag)*

*{*

*case 0:*

*//静止状态*

*initGlobalVariables();*

*break;*

*case 1://打招呼*

*initGlobalVariables();*

*Greet(currentTime, body\_center\_x, body\_center\_y, body\_center\_z,*

*left\_upper\_arm\_swing\_angle, right\_upper\_arm\_swing\_angle,*

*left\_lower\_arm\_swing\_angle, right\_lower\_arm\_swing\_angle,*

*left\_upper\_leg\_swing\_angle, right\_upper\_leg\_swing\_angle,*

*left\_lower\_leg\_swing\_angle, right\_lower\_leg\_swing\_angle);*

*break;*

*case 2://摊手*

*initGlobalVariables();*

*SpreadHands(currentTime, body\_center\_x, body\_center\_y, body\_center\_z,*

*left\_upper\_arm\_swing\_angle, right\_upper\_arm\_swing\_angle,*

*left\_lower\_arm\_swing\_angle, right\_lower\_arm\_swing\_angle,*

*left\_upper\_leg\_swing\_angle, right\_upper\_leg\_swing\_angle,*

*left\_lower\_leg\_swing\_angle, right\_lower\_leg\_swing\_angle);*

*break;*

*case 3://原地踏步*

*initGlobalVariables();*

*MarchOnTheSpot(currentTime, body\_center\_x, body\_center\_y, body\_center\_z,*

*left\_upper\_arm\_swing\_angle, right\_upper\_arm\_swing\_angle,*

*left\_lower\_arm\_swing\_angle, right\_lower\_arm\_swing\_angle,*

*left\_upper\_leg\_swing\_angle, right\_upper\_leg\_swing\_angle,*

*left\_lower\_leg\_swing\_angle, right\_lower\_leg\_swing\_angle);*

*break;*

*case 4://扎马步*

*initGlobalVariables();*

*HorseRidingStance(currentTime, body\_center\_x, body\_center\_y, body\_center\_z,*

*left\_upper\_arm\_swing\_angle, right\_upper\_arm\_swing\_angle,*

*left\_lower\_arm\_swing\_angle, right\_lower\_arm\_swing\_angle,*

*left\_upper\_leg\_swing\_angle, right\_upper\_leg\_swing\_angle,*

*left\_lower\_leg\_swing\_angle, right\_lower\_leg\_swing\_angle);*

*break;*

*case 5://自旋*

*//initGlobalVariables();*

*break;*

*default:*

*break;*

*}*

*glutPostRedisplay(); // 重新绘制场景*

*glutTimerFunc(1000 / 60, update, value + 1); // 设置下一帧更新时间*

*}*