# 计算机图形学仿真机器人

**实**

**验**

**报**

**告**

**学校：** 郑州大学

**院系**：信息工程学院

**专业**：软件工程

**班级**：卓越2班

**姓名**： 代东明

**学号**： 20162430804

**指导老师：**马岭

一、**实验目的**

熟悉视点观察函数的设置和使用

熟悉3D图形变换的设置和使用

进一步熟悉基本3D图形的绘制

体验透视投影和正面投影的不同效果

掌握简单机器人的编程

**二、实验设备**

装载windows操作系统的联想电脑，

Visual studio2017 软件

**三、实验内容**

打开vs，创建控制台项目，编写简单机器人程序

机器人包括以下几个部分

头：设计成球状，可进行顺时针，逆时针旋转

肩膀：设计成同心圆形状

腰部和肩膀之间的身体：设计成圆台形

双臂：可进行顺时针，逆时针旋转，同时手，小臂，也可单独旋转

腰部：同样用圆台代替

腿：和手臂相似，可分为大腿，小腿，可左右旋转， 上下旋转

机器人能够在界面内自由行走，身体各部分可以自由转动

**1）主要函数及解释：**

glNewList(HEAD, GL\_COMPILE);// //用于创建一个显示列表函数原型

glColor3f(1, 1, 1);设置颜色

glPushMatrix();//在变换完毕后用glPopMatrix()记录变换后的位置

gluSphere(qObj, 10, 10, 10); //（对象，球半径，Z片数,Y片数）

glRotatef(-turnHead, 0, 1, 0); //（旋转角度，轴向量）

glTranslatef(0, -10, 0); //平移函数（xyz）

glRotatef(90, 1, 0, 0);

gluCylinder(qObj, 5, 5, 5, 30, 30); //圆柱（对象，底半径，顶半径，\*\*，\*\*）

gluDisk(qObj, 0, 5, 30, 30); /\*同心圆（二次对象，内部半径，外部半径，围绕z轴的细分数， 关于圆盘被细分为原点的同心圆环的数目）\*/

glPopMatrix();//调用时返回原态

glEndList();//替换一个显示列表函数原型

glEnable()//启用功能，具体启用对象有括号内的参数决定

/\*在图像绘制的所有操作之后，要加入 glutPostRedisplay() 来重绘图像，

否则图像只有响应鼠标或键盘消息时才会更新图像。

有点像窗口重绘函数一样。它要求当前的回调函数返回时执行显示回调函数\*/

glClearDepth(2.0);//缓冲深度值

glMatrixMode(GL\_PROJECTION);//参数投影

/\*在要做下一步之前告诉计算机我要对“什么”进行操作了

要对投影相关进行操作，也就是把物体投影到一个平面上，就像我们照相一样，把3维 物体投到2维的平面上。

这样，接下来的语句可以是跟透视相关的函数

\*/

glLoadIdentity();

/\*无论以前进行了多少次矩阵变换，在该命令执行后，当前矩阵均恢复成一个单位矩阵，

即相当于没有进行任何矩阵变换状态。\*/

gluLookAt(490, 450, 0,/\*相机在世界坐标的位置\*/ 0, -85, myrobot.pos[1],/\*相机镜头对准的物体在世界坐标的位置\*/

0, 1, 0/\*头顶朝向的方向\*/);

glViewport(0, 0, w, h);

/\*把视景体截取的图像按照怎样的高和宽显示到屏幕上

前两个参数所在坐标系的原点位于当前窗口的左下角，即整个窗口位于第一象限内，一般情况，前两个参数都是0,0

后两个参数表示该视口的宽度和高度，可以把视口看做是一个矩形方框。

\*/

glutInit(&argc, argv);//传递一个正确的入口参数，使它能够正确地初始化OpenGL环境

glutInitWindowSize(800, 700);//设置窗口宽，高

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DEPTH | GLUT\_ACCUM | GLUT\_DOUBLE);

//设置图形显示模式

/\*GLUT\_RGBA：当未指明GLUT - RGBA或GLUT - INDEX时，是默认使用的模式。表明欲建立RGBA模式的窗口。

 GLUT\_DEPTH：使用深度缓存

GLUT\_ACCUM：让窗口使用累加的缓存。

GLUT\_DOUBLE：使用双缓存。以避免把计算机作图的过程都表现出来，或者为了平滑地实现动画。

(void)glutCreateWindow("Made by ddm");

/\*产生一个顶层的窗口。name 作为窗口的名字，也就是窗口标题栏显示的内容

返回值是生成窗口的标记符，可用函数glutGetWindow()加以引用

\*/

myinit();//初始化 准备函数

glutDisplayFunc(redraw);//调用redraw函数

glutReshapeFunc(reshape);//重新绘图，调用reshape

glutKeyboardFunc(key);//注册当前窗口的键盘回调函数 调用key

glLightfv(GL\_LIGHT0, GL\_POSITION, lightpos);

/\*glLightfv（光源编号，光源特性，参数数据）来设置光源

\*/

glutMainLoop();//循环

**2）、功能键介绍：**

功能键 Esc 退出 m 改变 机器人显示模式 d逆时针 转体 a 顺时针转体 w 向前步行 s后退

大写代表顺时针：

（h逆时针转动头部 j逆时针转动肩膀 k逆时针转动腰部及以下大写时顺时针转动

右臂 u 逆时针转动整条手臂 i 逆时针转动小手臂 o逆时针转动手爪

左臂 v顺时针转动整条手臂 b顺时针转动小手臂 n逆时针转动手爪

左腿 r逆时针转动整条腿 t逆时针转动小腿

右腿 d逆时针转动整条腿 f逆时针转动小腿 ）

**3）、主要代码：**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*头部编码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void head()//头

{

glNewList(HEAD, GL\_COMPILE);// //用于创建一个显示列表函数原型

glColor3f(1, 1, 1);

glPushMatrix();//在变换完毕后用glPopMatrix()记录变换后的位置

gluSphere(qObj, 10, 10, 10); //（对象，球半径，Z片数,Y片数）

glRotatef(-turnHead, 0, 1, 0); //（旋转角度，轴向量）

glTranslatef(0, -10, 0); //平移函数（xyz）

glRotatef(90, 1, 0, 0);

gluCylinder(qObj, 5, 5, 5, 30, 30); //圆柱（对象，底半径，顶半径，\*\*，\*\*）

gluDisk(qObj, 0, 5, 30, 30); /\*同心圆（二次对象，内部半径，外部半径，围绕z轴的细分数，关于 圆盘被细分为原点的同心圆环的数目）\*/

glPopMatrix();//调用时返回原态

glEndList();//替换一个显示列表函数原型

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*肩膀编码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void shouder()

{

glNewList(SHOUDER, GL\_COMPILE);

glColor3f(0, 0.8, 1);

glPushMatrix();//在变换完毕后用glPopMatrix()记录变换后的位置

glRotatef(90, 1, 0, 0);//旋转

gluCylinder(qObj, 8, 25, 8, 30, 30);//圆柱

gluDisk(qObj, 0, 8, 30, 30);//同心圆

glRotatef(-90, 1, 0, 0);

glTranslatef(0, -8, 0);//平移

glRotatef(90, 1, 0, 0);

gluDisk(qObj, 0, 25, 30, 30);

glPopMatrix();//调用时返回原态

glColor3f(1, 1, 1);

glEndList();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*腰部和肩膀之间身体编码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void body()

glNewList(BODY, GL\_COMPILE);//在变换完毕后用glPopMatrix()记录变换后的位置

glColor3f(0.8, 1, 0);

gluCylinder(qObj, 6, 15, 25, 30, 30);

glColor3f(1, 1, 1);

glEndList();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*腰部编码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void waist()

{

glNewList(WAIST, GL\_COMPILE);

glColor3f(0, 0, 1);

glPushMatrix();

glRotatef(90, 1, 0, 0);

gluCylinder(qObj, 6, 6, 2, 30, 30);

glRotatef(-90, 1, 0, 0);

glTranslatef(0, -2, 0);

glRotatef(90, 1, 0, 0);

gluCylinder(qObj, 6, 10, 10, 10, 10);

glRotatef(-90, 1, 0, 0);

glTranslatef(0, -10, 0);

glRotatef(90, 1, 0, 0);

gluDisk(qObj, 0, 10, 30, 30);

glPopMatrix();

glColor3f(0, 1, 1);

glEndList();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*小腿编码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void lowleg()

{

glColor3f(0.5, 0, 1);

glRotatef(90, 1, 0, 0);

gluCylinder(qObj, 3, 3, 50, 30, 30);

glRotatef(-90, 1, 0, 0);

glTranslatef(-3, -50, 0);

glRotatef(90, 0, 1, 0);

gluCylinder(qObj, 3, 3, 6, 30, 30);

glRotatef(-90, 0, 1, 0);

glTranslatef(3, -2, 6);

glPushMatrix();

if (!ileg % 2)

{glRotatef(turnFootR, 1, 0, 0);

}

else

{glRotatef(turnFootL, 1, 0, 0);

}

glPopMatrix();}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*大腿编码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void upleg()

{

glPushMatrix();

for (ileg = 0; ileg < 2; ileg++)

{

if (ileg % 2)

glTranslatef(-12, 0, 0);

glColor3f(0, 0.5, 1);

gluSphere(qObj, 4, 30, 30); //（对象，球半径，Z片数, Y片数）

glPushMatrix();

if (!ileg % 2)

glRotatef(turnUpLegR, 1, 0, 0);

else

glRotatef(turnUpLegL, 1, 0, 0);

if (!ileg % 2)

glRotatef(5, 0, 0, 1);

else

glRotatef(-5, 0, 0, 1);

glRotatef(90, 1, 0, 0);

gluCylinder(qObj, 4, 4, 20, 30, 30);

glRotatef(-90, 1, 0, 0);

glTranslatef(-4, -23, 0);

glRotatef(90, 0, 1, 0);

gluCylinder(qObj, 3, 3, 8, 30, 30);

glRotatef(-90, 0, 1, 0);

glTranslatef(4, 0, 0);

glColor3f(1, 0, 1);

glPushMatrix();

if (!ileg % 2)

{glRotatef(turnMiddleLegR, 1, 0, 0);

lowleg();

}

else

{ glRotatef(turnMiddleLegL, 1, 0, 0);

lowleg();

glPopMatrix();

glPopMatrix();

}

glPopMatrix();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*手部编码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void hand()

{

glPushMatrix();

glTranslatef(-1, -6, -6);

glRotatef(90, 0, 1, 0);

drawTorus();//调用手爪

glPopMatrix();//调用时返回原态

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*小臂\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void lowarm()

{

glRotatef(90, 1, 0, 0);

gluCylinder(qObj, 3, 3, 20, 30, 30);

glRotatef(-90, 1, 0, 0);

glTranslatef(-2, -20, 0);

glRotatef(90, 0, 1, 0);

gluCylinder(qObj, 3, 3, 4, 30, 30);

glRotatef(-90, 0, 1, 0);

glTranslatef(3, -2, 6);

glPushMatrix();

if (!iarm % 2)

{glRotatef(turnHandR, 1, 0, 0);

hand();}

else

{ glRotatef(turnHandL, 1, 0, 0);

hand();

if (!cancatchball)

{glTranslatef(-1, -6, -6);

}}

glPopMatrix();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*胳膊\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void uparm()

{ glPushMatrix();

for (iarm = 0; iarm < 2; iarm++)

{ if (iarm % 2)

glTranslatef(-50, 0, 0);

glColor3f(1, 0, 1);

gluSphere(qObj, 5, 30, 30);

glPushMatrix();

if (!iarm % 2)

glRotatef(turnUpArmR, 1, 0, 0);

else

glRotatef(turnUpArmL, 1, 0, 0);

if (!iarm % 2)

glRotatef(10, 0, 0, 1);

else

glRotatef(-10, 0, 0, 1);

glRotatef(90, 1, 0, 0);

gluCylinder(qObj, 4, 4, 20, 30, 30);

glRotatef(-90, 1, 0, 0);

glTranslatef(-4, -23, 0);

glRotatef(90, 0, 1, 0);

gluCylinder(qObj, 3, 3, 8, 30, 30);

glColor3f(0, 1, 0.3);

glRotatef(-90, 0, 1, 0);

glTranslatef(4, 0, 0);

glPushMatrix();

if (!iarm % 2)

{ glRotatef(turnMiddleArmR, 1, 0, 0);

lowarm();}

else

{ glRotatef(turnMiddleArmL, 1, 0, 0);

lowarm(); }

glPopMatrix();

glPopMatrix(); }

glPopMatrix(); }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*初始化机器人对象\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void myinit()//初始化

{ qObj = gluNewQuadric();//创建二次对象

glEnable(GL\_DEPTH\_TEST);//参数 跟据函数glAlphaFunc的条件要求来决定图形透明的层度是否显示

glEnable(GL\_COLOR\_MATERIAL);//启用每一像素的色彩为位逻辑运算

// 用于启用各种功能。功能由参数决定

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

//init robot pos

myrobot.dir[0] = 0;

myrobot.dir[1] = 0;

myrobot.pos[0] = 0;

myrobot.pos[1] = 0;

//Robot

head();

shouder();

body();

waist();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*对于机器人动作作图\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void drawRobot()//画图

{

glPushMatrix();

glPushMatrix();

glRotatef(-90, 1, 0, 0);

glCallList(BODY);

glPopMatrix();//一次动作

glPushMatrix();

glTranslatef(0, 33, 0);

glRotatef(turnShouder, 0, 1, 0);

glCallList(SHOUDER);

glTranslatef(25, -7, 0);

uparm();

glPopMatrix();//二次变换

glPushMatrix();

glTranslatef(0, 45, 0);

glRotatef(turnHead, 0, 1, 0);

glCallList(HEAD);

glPopMatrix();//三次变换

glPushMatrix();

glRotatef(turnWaist, 0, 1, 0);

glCallList(WAIST);

glTranslatef(6, -15, 0);

upleg();

glPopMatrix();//4次变换

glPopMatrix();//收尾

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*功能键设定\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void key(unsigned char key, int x, int y)//功能键

{if (iskick == 2)

{iskick = 1; }

if (key == '\033')

exit(0);//Esc退出

if (key == 'm') // m 改变模式

mode++;

//direction

if (key == 'd')// d 逆时针转体

{turnbody += 10;}

else if (key == 'a') // 顺时针转体

{ turnbody -= 10;

}

myrobot.dir[0] = sin(M\_PI / 180 \* turnbody);

myrobot.dir[1] = cos(M\_PI / 180 \* turnbody);

if (key == 'w')//向前走

{ iswalk = 1;

float lastpos0 = myrobot.pos[0];

float lastpos1 = myrobot.pos[1];

myrobot.pos[0] += 5 \* myrobot.dir[0];

myrobot.pos[1] += 5 \* myrobot.dir[1];

//坐标前进

//border deal

if (myrobot.pos[0] >= 450 ||

myrobot.pos[0] <= -450 ||

myrobot.pos[1] <= -450 ||

myrobot.pos[1] >= 450//wall)

{ myrobot.pos[0] = lastpos0;

myrobot.pos[1] = lastpos1;}//更新坐标

robotrun();//调转 至动画 }

if (key == 's')

{ iswalk = 1;

float lastpos0 = myrobot.pos[0];

float lastpos1 = myrobot.pos[1];

myrobot.pos[0] -= 5 \* myrobot.dir[0];

myrobot.pos[1] -= 5 \* myrobot.dir[1];//坐标后移

//border deal

if (myrobot.pos[0] >= 450 ||

myrobot.pos[0] <= -450 ||

myrobot.pos[1] <= -450 ||

myrobot.pos[1] >= 450//wall )

{ myrobot.pos[0] = lastpos0;

myrobot.pos[1] = lastpos1; }//更新坐标

robotrun(); }

if (key == 'h')

{ turnHead += 10;

} // h逆时针转动头部

else if (key == 'j')

{ turnShouder += 10;

}// j 逆时针转动肩膀

else if (key == 'k')

{ turnWaist += 10; } // k 逆时针转动 腰部以下

else if (key == 'H')

{ turnHead -= 10; }//H 顺时针转动头部

else if (key == 'J')

{ turnShouder -= 10; } //J 逆时针转动肩膀

else if (key == 'K')

{ turnWaist -= 10; }// K 逆时针转动 腰部以下

if (key == 'u')

{ turnUpArmR += 10; } //逆时针转动整个手臂 u

else if (key == 'i')

{ turnMiddleArmR += 10; } // i 逆时针转动小臂

else if (key == 'o')

{ turnHandR += 10; }// 逆时针 转手爪子 o

else if (key == 'U')

{ turnUpArmR -= 10; }

else if (key == 'I')

{ turnMiddleArmR -= 10; }

else if (key == 'O')

{ turnHandR -= 10; }

if (key == 'v')

{ turnUpArmL += 10;}

else if (key == 'b')

{ turnMiddleArmL += 10; }

else if (key == 'n')

{ turnHandL += 10; }

else if (key == 'V')

{ turnUpArmL -= 10; }

else if (key == 'B')

{ turnMiddleArmL -= 10; }

else if (key == 'N')

{ turnHandL -= 10; }

if (key == 'r')

{ turnUpLegR += 10; } //逆时针转动右腿

else if (key == 't')

{ turnMiddleLegR += 10;}//逆时针转动右小腿

else if (key == 'R')

{ turnUpLegR -= 10; }

else if (key == 'T')

{ turnMiddleLegR -= 10; }

if (key == 'l')

{ turnUpLegL += 10;}

else if (key == 'f')

{turnMiddleLegL += 10;}//逆时针左小腿 f

else if (key == 'L')

{turnUpLegL -= 10; }

else if (key == 'F')

{ turnMiddleLegL -= 10;}

glutPostRedisplay();

}

void redraw(void)//作图

{ //设置清除屏幕的颜色，并清除屏幕和深度缓冲

glClearColor(1.0, 1.0, 0.5, 1.0);//

glClearDepth(2.0);//缓冲深度值

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

//

//变换并绘制物体

glMatrixMode(GL\_PROJECTION);//参数投影

/\*在要做下一步之前告诉计算机我要对“什么”进行操作了

要对投影相关进行操作，也就是把物体投影到一个平面上，就像我们照相一样，把3维物体投到2维的平面上。

这样，接下来的语句可以是跟透视相关的函数

\*/

glLoadIdentity();

/\*无论以前进行了多少次矩阵变换，在该命令执行后，当前矩阵均恢复成一个单位矩阵，

即相当于没有进行任何矩阵变换状态。\*/

// gluPerspective(80.0, 1, 1,3000.0);

gluPerspective(75.0, 2, 2.5, 3000.0);//（角度，宽高比，沿z轴方向的两裁面之间的距离的近处，沿z轴方向的两裁面之间的距离的远处）

/\*\*改变图形胖瘦，视图\*\*\*\*\*\*\*\*\*\*\*/

glPushMatrix();//记录每次变换

glLoadIdentity();

/\*每次建立顶点的时候，都是以当前物体的中心作为世界的原点的

（换句话说，也就是上一轮物体变换之后，原点的位置作为新的物体的中心）\*/

glOrtho(0, 100, 0, 100, -1, 1);//（左，右，底，顶，近，远）

/\*平行修剪空间，无论物体距离相机多远，投影后的物体大小尺寸不变\*/

glMatrixMode(GL\_MODELVIEW);

/\*功能是指定哪一个矩阵是当前矩阵。创建一个投影矩阵并且与当前矩阵相乘，得到的矩阵设定为当前变换

GL\_MODELVIEW,对模型视景矩阵堆栈应用随后的矩阵操作.

GL\_PROJECTION,对投影矩阵应用随后的矩阵操作.

GL\_TEXTURE,对纹理矩阵堆栈应用随后的矩阵操作.\*/

glLoadIdentity();

glNormal3f(0, 0, 1);

glTranslatef(37, 37, 0);

// glTranslatef(37,37,600);

glFlush();

//用于强制刷新缓冲，保证绘图命令将被执行，而不是存储在缓冲区[2] 中等待其他的OpenGL命令。

glMatrixMode(GL\_PROJECTION);

glPopMatrix();

/\*回到glPushMatrix()之前的状态\*/

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(0, -8.6, -300);//你绘点坐标的原点在当前原点的基础上平移一个(x,y,z)向量。

if (mode % 2)

{

glDisable(GL\_LIGHTING);//使用当前光照计算顶点颜色。否则只是将当前的颜色和顶点简单的关联在一起

glDisable(GL\_DEPTH\_TEST);//进行深度比较和更新深度缓冲

glDisable(GL\_NORMALIZE);//在转换之后和光照之前将法线向量标准化成单位长度

/\*disable是关闭各种功能\*/

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

}/\*（确定显示模式将适用于物体的哪些部分控制多边形的正面和背面的绘图模式，

mode这个参数确定选中的物体的面以何种方式显示）

GL\_FRONT表示显示模式将适用于物体的前向面（也就是物体能看到的面）

GL\_LINE表示显示线段，多边形用轮廓显示\*/

else

{//启动模式

glEnable(GL\_LIGHTING);//启用灯源

glEnable(GL\_DEPTH\_TEST);//启用深度测试根据坐标的远近自动隐藏被遮住的图形（材料）

glEnable(GL\_NORMALIZE);//根据函数glNormal的设置条件，启用法向量

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

}

glLoadIdentity();

//gluLookAt(490,450,0,0,-85,myrobot.pos[1],0,1,0);

gluLookAt(490, 450, 0,/\*相机在世界坐标的位置\*/ 0, -85, myrobot.pos[1],/\*相机镜头对准的物体在世界坐标的位置\*/

0, 1, 0/\*头顶朝向的方向\*/);

/\*该函数定义一个视图矩阵，并与当前矩阵相乘。

\*/

glPushMatrix();

//glTranslatef(0,-8.6,0);

glTranslatef(0, 300, 0);

glRotatef(turnleft, 0, 1, 0);

glRotatef(turnupward, 1, 0, 0);

glRotatef(turnfront, 0, 0, 1);

glPushMatrix();

//glTranslatef(0,-200,0);

glTranslatef(0, elevation, 0);

//Robot

glPushMatrix();

if (iswalk)

glTranslatef(myrobot.pos[0], 0, myrobot.pos[1]);

glRotatef(turnbody, 0, 1, 0);

drawRobot();

glPopMatrix();

glPopMatrix();

glFlush();

glutSwapBuffers();

/\*交换前台后台缓冲区指针，解决了频繁刷新导致的画面闪烁问题\*/

}

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

{/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*输出各键代表的功能\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glutInit(&argc, argv);//传递一个正确的入口参数，使它能够正确地初始化OpenGL环境。

glutInitWindowSize(800, 700);//设置窗口宽，高

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DEPTH | GLUT\_ACCUM | GLUT\_DOUBLE);

//设置图形显示模式

/\*GLUT\_RGBA：当未指明GLUT - RGBA或GLUT - INDEX时，是默认使用的模式。表明欲建立RGBA模式的窗口。

 GLUT\_DEPTH：使用深度缓存

GLUT\_ACCUM：让窗口使用累加的缓存。

GLUT\_DOUBLE：使用双缓存。以避免把计算机作图的过程都表现出来，或者为了平滑地实现动画。

(void)glutCreateWindow("Made by ddm");

/\*产生一个顶层的窗口。name 作为窗口的名字，也就是窗口标题栏显示的内容

返回值是生成窗口的标记符，可用函数glutGetWindow()加以引用

myinit();//初始化 准备函数

glutDisplayFunc(redraw);//调用redraw函数

glutReshapeFunc(reshape);//重新绘图，调用reshape

glutKeyboardFunc(key);//注册当前窗口的键盘回调函数 调用key

glMatrixMode(GL\_MODELVIEW/\*对模型视景的操作\*/);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

/\*启动\*/

glLightfv(GL\_LIGHT0, GL\_POSITION, lightpos);

/\*glLightfv（光源编号，光源特性，参数数据）来设置光源

\*/

glutMainLoop();//循环

return 0;

}

**四、实验总结**

1、进行3D机器人仿真实验，需要对作图函数进行熟练掌握，而且实验中的许多函数多我们来说不是很熟悉，甚至有些函数中参数不同，函数的意义就完全不同，因此，熟练掌握函数，参数及其使用与配合是及其重要的。

2、编程中要构造许多功能，或者作许多图，但是这些图很多都用到了相同的函数，只是组合不同，展示出来的效果也不同，因此需要掌握不同函数之间的搭配。

3、由于机器人编程中要设置的功能，部位太多，因此需要设置很多函数，由此编程时容易混乱，因此在原程序中做好注释，使层次清晰，易于理解。

**如下图：机器人图形展示**

