**山 东 大 学**

**学 生 实 验 报 告**

**学 院：** 计算机科学与技术

**课程名称：** 计算机图形学

**专业班级：** 2014计基地

**学 号：** 201400301202

**姓 名：** 王瑶

**学生实验报告(一)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 学生姓名 | 王瑶 | | 学号 | | 201400301202 | 同组人： 无 |
| 实验项目 | **Digital Image Fundamentals** | | | | |
| ■必修 □选修 | | □演示性实验 □验证性实验 ■操作性实验 □综合性实验 | | | | |
| 实验地点 |  | | | 实验仪器台号 | |  |
| 指导教师 |  | | | 实验日期及节次 | |  |

1. **实验综述**
2. 实验目的及要求

使机器人像人一样行走。

1. 实验仪器、设备或软件

VS2015+openGL

1. **算法分析**

机器人外形用立方体表示，关节处用圆球表示，机器人关节的变换用opengl自带的旋转函数，设置好每个部位的旋转中心。要想使得机器人行走起来，使用opengl自带的平移函数，控制头、身体、左臂上关节、右臂上关节、左腿上关节、右腿上关节同步平移。

**三、实验过程（实验步骤、记录、数据、分析）**

1、实验代码粘贴：

#include<windows.h>

#include<gl/glut.h>

#pragma comment(lib, "glut32.lib")

#include <iostream>

using namespace std;

GLfloat no\_mat[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat no\_shininess[] = { 0.5 };

GLfloat low\_shininess[] = { 5.0 };

GLfloat high\_shininess[] = { 100.0 };

GLfloat mat\_emission[] = { 0.3, 0.2, 0.2, 0.0 };

static float yy = 0.0;

static int angle = 0, lshoulder = 0, lelbow = 0, rshoulder = 0, relbow = 0,

lhips = 0, rhips = 0, lfoot = 0, rfoot = 0, flagneck = 0, flaglshoulder = 0,

flaglelbow = 0, flagrshoulder = 0, flagrelbow = 0, flaglhips = 0,

flagrhips = 0, flaglfoot = 0, flagrfoot = 0;

void init(void)

{

glClearColor(0, 0, 0, 0);

glShadeModel(GL\_SMOOTH);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

}

void draw\_head(void)

{

glPushMatrix();

glTranslatef(0, 3.5, yy);

glTranslatef(0, 1, 0);

glutSolidCube(2);

glPopMatrix();

}

void draw\_body(void)

{

glPushMatrix();

glTranslatef(0, 1.5, yy);

glScalef(0.5, 1, 0.4);

glutSolidCube(4);

glPopMatrix();

}

void draw\_leftshoulder(void)

{

glPushMatrix();

glTranslatef(1.5, 3, yy);

glRotatef(lshoulder, 1, 0, 0);

glTranslatef(0, -0.5, 0);

glScalef(0.4, 1, 0.5);

glutSolidCube(2);

glScalef(1 / 0.4, 1 / 1, 1 / 0.5);

glTranslatef(0, -1.4, 0);

glRotatef(lelbow, 1, 0, 0);

glutWireSphere(0.4, 200, 500);

glScalef(0.4, 1, 0.5);

glTranslatef(0, -1.4, 0);

glutSolidCube(2);

glPopMatrix();

}

void draw\_rightshoulder(void)

{

glPushMatrix();

glTranslatef(-1.5, 3, yy);

glRotatef(rshoulder, 1, 0, 0);

glTranslatef(0, -0.5, 0);

glScalef(0.4, 1, 0.5);

glutSolidCube(2);

glScalef(1 / 0.4, 1 / 1, 1 / 0.5);

glTranslatef(0, -1.4, 0);

glRotatef(relbow, 1, 0, 0);

glutWireSphere(0.4, 200, 500);

glScalef(0.4, 1, 0.5);

glTranslatef(0, -1.4, 0);

glutSolidCube(2);

glPopMatrix();

}

void draw\_leftfoot(void)

{

glPushMatrix();

glTranslatef(-0.6, -0.6, yy);

glRotatef(lfoot, 1, 0, 0);

glTranslatef(0, -1, 0);

//缩放

glScalef(0.4, 1, 0.5);

glutSolidCube(2);

glScalef(1 / 0.4, 1 / 1, 1 / 0.5);

glTranslatef(0, -1.4, 0);

glRotatef(lhips, 1, 0, 0);

glutWireSphere(0.4, 200, 500);

glScalef(0.4, 1, 0.5);

glScalef(1, 1, 1);

glTranslatef(0, -1.4, 0);

glutSolidCube(2);

glPopMatrix();

}

void draw\_rightfoot(void)

{

glPushMatrix();

glTranslatef(0.6, -0.6, yy);

glRotatef(rfoot, 1, 0, 0);

glTranslatef(0, -1, 0);

glScalef(0.4, 1, 0.5);

glutSolidCube(2);

glScalef(1 / 0.4, 1 / 1, 1 / 0.5);

glTranslatef(0, -1.4, 0);

glRotatef(rhips, 1, 0, 0);

glutWireSphere(0.4, 200, 500);

glScalef(0.4, 1, 0.5);

glScalef(1, 1, 1);

glTranslatef(0, -1.4, 0);

glutSolidCube(2);

glPopMatrix();

}

void display(void)

{

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

glPushMatrix();

glRotatef(angle, 0, 1, 0);

glTranslatef(0, 4, 0);

draw\_body();

draw\_head();

draw\_leftshoulder();

draw\_rightshoulder();

draw\_leftfoot();

draw\_rightfoot();

glPopMatrix();

glutSwapBuffers();

}

void reshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-8, 8, -8, 8, -10, 10);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(0.0, 4.0, 5, 0.0, 4, 0.0, 0.0, 1.0, 0.0);

}

void keyboard(unsigned char key, int x, int y)

{

switch (key)

{

case '0':

angle = (angle + 5) % 360;

glutPostRedisplay();

break;

case '1':

angle = (angle - 5) % 360;

glutPostRedisplay();

break;

case '2':

if (yy < 8.0) {

yy += 0.05;

}

else {

yy = -8.0;

}

if (flaglshoulder)

{

lshoulder += 5;

if (lshoulder >= 60)flaglshoulder = 0;

}

else

{

lshoulder -= 5;

if (lshoulder <= -60)flaglshoulder = 1;

}

glutPostRedisplay();

//3

if (flagrshoulder)

{

rshoulder -= 5;

if (rshoulder <= -60)flagrshoulder = 0;

}

else

{

rshoulder += 5;

if (rshoulder >= 60)flagrshoulder = 1;

}

glutPostRedisplay();

//4

if (flaglelbow)

{

lelbow += 2.5;

if (lelbow >= 0)flaglelbow = 0;

}

else

{

lelbow -= 2.5;

if (lelbow <= -30)flaglelbow = 1;

}

glutPostRedisplay();

//5

if (flagrelbow)

{

relbow += 2.5;

if (relbow >= 0)flagrelbow = 0;

}

else

{

relbow -= 2.5;

if (relbow <= -30)flagrelbow = 1;

}

glutPostRedisplay();

//6

if (flaglhips)

{

lhips += 2.5;

if (lhips >= 30)flaglhips = 0;

}

else

{

lhips -= 2.5;

if (lhips <= 0)flaglhips = 1;

}

glutPostRedisplay();

//7

if (flagrhips)

{

rhips += 2.5;

if (rhips >= 30)flagrhips = 0;

}

else

{

rhips -= 2.5;

if (rhips <= 0)flagrhips = 1;

}

glutPostRedisplay();

//8

if (flaglfoot)

{

lfoot += 5;

if (lfoot >= 60)flaglfoot = 0;

}

else

{

lfoot -= 5;

if (lfoot <= -60)flaglfoot = 1;

}

glutPostRedisplay();

//9

if (flagrfoot)

{

rfoot -= 5;

if (rfoot <= -60)flagrfoot = 0;

}

else

{

rfoot += 5;

if (rfoot >= +60)flagrfoot = 1;

}

glutPostRedisplay();

break;

default:

break;

}

}

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

{

printf("按键‘0’,‘1’为整体旋转\n");

printf("按键‘2’使机器人行走\n");

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RED);

glutInitWindowSize(600, 600);

glutInitWindowPosition(100, 10);

glutCreateWindow(argv[0]);

init();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutMainLoop();

return 0;

}

2、实验结论：

要想机器人像人一样行走不会散架，要合理的运用opengl的旋转函数和平移函数。对机器人的头、身体、左右手臂、左右腿分层次进行操作，其中，左手臂分为上手臂、关节和下手臂，细节到机器人的左手臂运动时，再单独对上手臂、关节和下手臂进行旋转设置，同理，处理右手臂和两条腿。设置手臂和腿的旋转角度，当达到设置的极值角度时，使手臂和腿反向旋转。设置机器人的行走路程，当达到一定的坐标时，使机器人从屏幕的另一个方向进入屏幕继续行走。

**四、指导教师评语及成绩：**

评语：

成绩： 指导教师签名：

批阅日期：