天津大學

图形学实验报告

—— 利用 OpenGL 实现动态的安卓机器人模型实验

学　　院 计算机科学与技术

专　　业 计算机科学与技术

年　　级 2012

姓　　名 王雨滕

学　　号 3012216083

2014 年 11 月 1 日

一、 实验的目的与要求

1. 了解 OpenGL，初步掌握 OpenGL 图形开发流程
2. 调用 OpenGL 库函数绘制 3D 图形，并组合成安卓机器人模型
3. 实现基本的变换操作，使机器人动起来
4. 添加光照处理，设置光源和材质属性

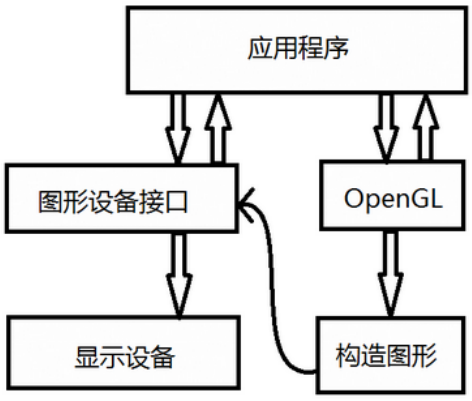
二、 主要技术及实现

1. OpenGL 环境的搭建：

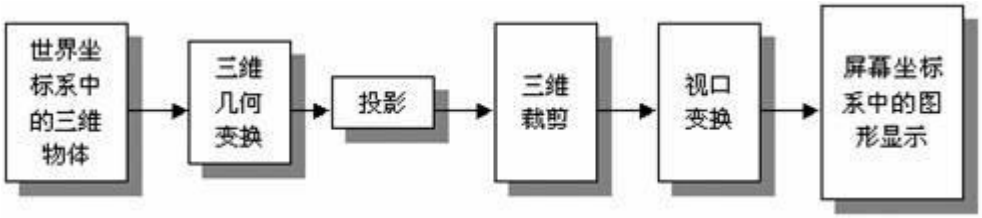
   1) 编译环境：Win8.1+Visual Studio2013+OpenGL
   2) 实验采用非 MFC 开发环境进行搭建。
      - ✧ 将下载的 GLUT 工具包压缩包解开，将得到 5 个文件
      - ✧ 将解压得到的 glut.h 放入
        C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\include\GL\
      - ✧ 把解压得到的 glut.lib 和 glut32.lib 放入
        C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\lib
      - ✧ 把解压得到的 glut.dll 和 glut32.dll 放入
        C:\Windows\SysWOW64 (windows8.1 64 位操作系统)
   3) 建立一个 OpenGL 工程。
      - ✧ 创建一个 Win32 Console Application。（工程名为 simpleROT）
      - ✧ 选择 ：
        project->project property-> Configuration Properties->Linker->Input->Additional Dependencies 在其中添加 opengl32.lib;glu32.lib;glut32.lib
      - ✧ 单击 Project Settings 中的 C/C++标签，将 Preprocessor definitions 中的_CONSOLE 改为__WINDOWS。

      注：编写 OpenGL 程序时需要添加头文件：#include <GL/glut.h>，并在其前面加上:
      ```
      #define GLUT_DISABLE_ATEXIT_GACK
      #include<windows.h>
      ```

2. OpenGL 图形开发流程：
   1) OpenGL 实现基本原理如下:



2) OpenGL 中三维物体的显示过程



3. 调用 OpenGL 库函数绘制机器人模型：

1) 绘制头：glutSolidSphere(0.1f, 100, 100);

2) 绘制长方形的身体。先绘制立方体，再将其缩放成长方体。

```
glScalef(0.3,0.4,0.2);  //模型变换成长方体
glutSolidCube(1.0); //绘制实心立方体
```

3) 绘制胳膊和腿。

下例为绘制左胳膊：先绘制圆柱，再将其旋转移动到机器人的位置。

```
glTranslatef(-0.2f, 0.5f, 0);//移动
glRotatef(75, 1.0f,0, 0 );//旋转
GLUquadric *pobj;
pobj = gluNewQuadric();
gluCylinder(pobj,0.05,0.05,0.3,20,30); //绘制圆柱
```

4) 设置机器人的颜色。采用 ub 做后缀的函数，以 255 表示最大的使用。

```
glColor3ub(146, 94, 162); //紫色
```

5) 绘制机器人的基本函数:

```
void myDisplay(void) {
glClear(GL_COLOR_BUFFER_BIT);
 /* 进行机器人的绘制 */
glEnd();
glFlush(); }
```

4. 实现基本的变换操作，使机器人动起来：

1) 加入数组，当按下'a'时角度递增。写入 KeyBoard 函数中

```
if (key = 'a') {
angx[0] +=5;
angx[0] = angx[0] % 360;
}
```

然后，用旋转、移动函数进行变换。

```
glPushMatrix();
glTranslatef(-0.1f, 0.1f, 0);
glRotatef(angx[4], 0.1f, 0, 0);
glTranslatef(0.1f,- 0.1f, 0);
leftLeg();
glPopMatrix();
```

其中，状态的保存很重要。用 glPushMatrix();glPopMatrix();矩阵堆栈保存状态：一是在构建层次模型，在进行绘制时可以回到先前的视图，而不需要我们重新计算绘制。

2) OpenGL 把三维坐标中的物体绘制到二维屏幕，绘制的顺序是按照代码的顺序来进行的。为了解决后绘制的物体会遮住先绘制的物体，使用深度测试可以解决这一问题。使用的方法是：

i).以 GL_DEPTH_TEST 为参数调用 glEnable 函数，启动深度测试。

ii).在必要时（通常是每次绘制画面开始时），清空深度缓冲。

合并写为：

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

✧ 注意：i.“先移动后旋转”和“先旋转后移动”得到的结果不同。

ii.在堆栈中，方法的执行顺序是从下到上。

iii.变换之前，需要在函数最前面进行深度测试: glEnable(GL_DEPTH_TEST);

5. 创建透视效果视图，进行投影和视图变换:

```
glMatrixMode(GL_PROJECTION); //投影
glLoadIdentity();
gluPerspective(90.0f, 1.0f, 1.0f, 20.0f);
glMatrixMode(GL_MODELVIEW); //视图变换
glLoadIdentity();
gluLookAt(0.8, 0.5, -1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
```

6. 添加光照处理，设置光源和材质属性并打开光照：

设置光源和材质的时候要注意设置环境反射、漫反射和镜面反射。

1) GL_AMBIENT、GL_DIFFUSE、GL_SPECULAR 属性。这三个属性表示了光源所发出的光的反射特性，每个属性由四个值表示，分别代表了颜色的 R, G, B, A 值。GL_AMBIENT 为环境光照，GL_DIFFUSE 为漫反射后得到的光强，GL_SPECULAR 镜面反射后得到的光强。 这三种属性是光源和材质所共有的。
2) GL_SHININESS 属性。又称“镜面指数”，范围为 0-128。值越小，表示材质越粗糙，点光源发射的光线照射到上面，可以产生较大亮点。
3) GL_EMISSION 属性。由四个值组成，表示一种材质微微的向外发射的颜色。
4) 用 glLight*函数设置光源的属性，glMaterial*函数设置材质的属性，glLightModel*函数设置光照模式。
5) 用 glEnable(GL_LIGHT0)，开启第 0 号光源，glDisable 函数关闭光源。

三、 实验结果

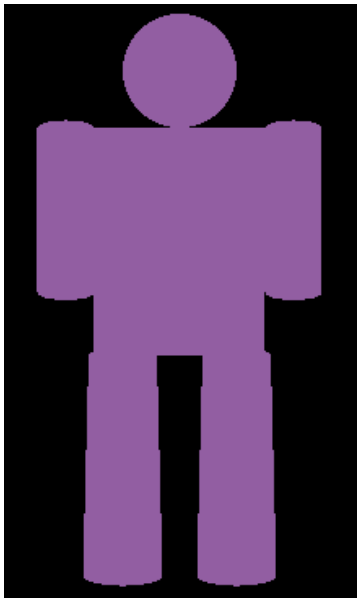


图 1:绘制成的机器人

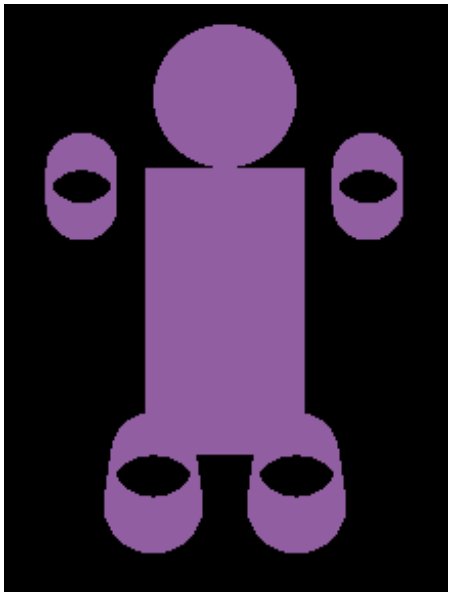


图 2 :可以动的机器人

图 3:变换视角观察到的机器人

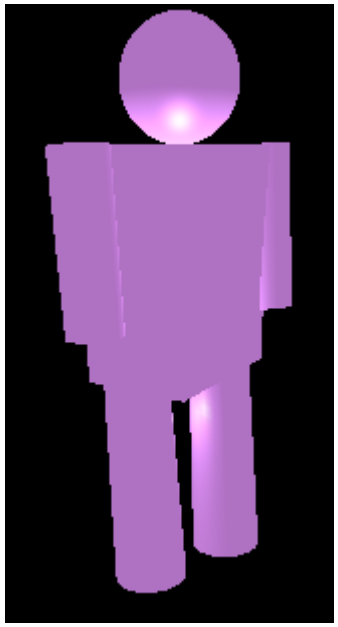


图 4:添加材质和光照后的机器人

四、 实验分析和总结

通过这次实验初步掌握了对 OpenGL 的基本使用，体会到了它最为一个 API 内封装函数的强大。通过从对一个食物完全一无所知到一步一步慢慢学习掌握的过程虽然艰难，但是完成之后会有莫名的成就感。每次图形学课实验都能够使我的自学能力大大的提升。

通过对机器人的绘制了解到一些基本几何图形的画法、基本的图形变换：位移、旋转、缩放、通过对世界坐标和物理坐标的应用和转换对绘图中的坐标系有了更深刻的理解。同时通过使机器人动起来学到了响应键盘的函数和控制能动的算法。最后通过添加光照和材质处理使自己的机器人更加逼真，也进一步理解了环境反射、漫反射和镜面反射。

笔者这次实验的不足之处是没有实现机器人的胳膊和腿分开动，按下 a 键之后,机器人的胳膊、身体和腿会一起进行旋转。这点的改进方法是应该加深对键盘读入函数的理解。我会继续学习和改进。

五、 核心源代码

```
#define GLUT_DISABLE_ATEXIT_HACK
#include <windows.h>
#include <GL/gl.h>
#include<GL/glut.h> //已包含gl.h和glu.h
#include<math.h>
#include<time.h>
#include<stdio.h>
#include<stdlib.h>

int angx[5] = { 0, 0, 0, 0, 0 };

//定义机器人光源，白色
GLfloat light_position[] = { 0.0f, 0.0f, 0.0f, 1.0f };//光源位置
GLfloat light_ambient[] = { 0.0f, 0.0f, 0.0f, 1.0f }; //环境光
GLfloat light_diffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };   //漫反射光 黑色
GLfloat light_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f }; //镜面反射光 黑色

//定义机器人的材质
GLfloat rot_position[] = { 0.0f, 0.0f, 0.0f, 1.0f };
GLfloat rot_ambient[] = { 146 / 255.0f, 94.0 / 255.0f, 162 / 255.0f, 1.0f };
GLfloat rot_diffuse[] = { 146 / 255.0f, 94.0 / 255.0f, 162 / 255.0f, 1.0f };
GLfloat rot_specular[] = { 146 / 255.0f, 94.0 / 255.0f, 162 / 255.0f, 1.0f };
GLfloat rot_emission[] = { 146 / 255.0f, 94.0 / 255.0f, 162 / 255.0f, 1.0f }; //材质本
身发光
GLfloat rot_shininess = 30.0f;//镜面指数，产生的亮点大小，0-128

//画出机器人的头
void Head(void)
{
    glColor3ub(146, 94, 162);
    glTranslatef(0, 0.6f, 0);
    glutSolidSphere(0.1f, 100, 100);//球
}
//画身体
void Body(void)
{

    glColor3ub(146, 94, 162);
    glTranslatef(0, 0.3f, 0);//移动
    glScalef(0.3,0.4,0.2);  //模型变换成长方体
    glutSolidCube(1.0); //绘制实心立方体
}
```

```
//左胳膊
void leftArm(void)
{
    glColor3ub(146, 94, 162);
    glTranslatef(-0.2f, 0.5f, 0);//移动
    glRotatef(75, 1.0f,0, 0 );//旋转
    GLUquadric *pobj;
    pobj = gluNewQuadric();
    gluCylinder(pobj,0.05,0.05,0.3,20,30); //绘制圆柱
}
//右胳膊
void rightArm(void)
{
    glColor3ub(146, 94, 162);
    glTranslatef(0.2f, 0.5f, 0);//移动
    glRotatef(75, 1.0f, 0, 0);//旋转
    GLUquadric *pobj;
    pobj = gluNewQuadric();
    gluCylinder(pobj, 0.05, 0.05, 0.3, 20, 30); //绘制圆柱
}
//左腿
void leftLeg(void)
{
    glColor3ub(146, 94, 162);
    glTranslatef(-0.1f, 0.1f, 0);
    glRotatef(75, 1.0f, 0, 0);
    GLUquadric *pobj;
    pobj = gluNewQuadric();
    gluCylinder(pobj,  0.06,0.07 ,0.4, 20, 30); //绘制实心立方体
}
//右腿
void rightLeg(void)
{
    glColor3ub(146, 94, 162);
    glTranslatef(0.1f, 0.1f, 0);//再移动
    glRotatef(75, 1.0f, 0, 0);//先旋转
    GLUquadric *pobj;
    pobj = gluNewQuadric();
    gluCylinder(pobj, 0.06,0.07, 0.4, 20, 30); //绘制实心立方体
}

void keyBoard(unsigned char key, int x, int y)
{
    if (key = 'a') {
```

```
            angx[0] +=5;
            angx[0] = angx[0] % 360;
        }
        if (key = 'b') {
            angx[1] += 5;
            angx[1] = angx[1] % 360;
        }
        if (key = 'c') {
            angx[2] += 5;
            angx[2] = angx[2] % 360;
        }
        if (key = 'd') {
            angx[3] += 5;
            angx[3] = angx[3] % 360;
        }
        if (key = 'e') {
            angx[4] += 5;
            angx[4] = angx[4] % 360;
        }
        if (key = 'f') {
            angx[5] += 5;
            angx[5] = angx[5] % 360;
        }
        glutPostRedisplay();
}
void  myDisplay(void)
{
        glEnable(GL_DEPTH_TEST);//深度测试
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);//清楚颜色深度缓存位



        //旋转左腿
        glPushMatrix();
        glTranslatef(-0.1f, 0.1f, 0);
        glRotatef(angx[4], 0.1f, 0, 0);
        glTranslatef(0.1f,- 0.1f, 0);
        leftLeg();
        glPopMatrix();

        //旋转右腿
        glPushMatrix();
        glTranslatef(0.1f, 0.1f, 0);
        glRotatef(angx[5], 0.1f, 0, 0);
```

```
    glTranslatef(-0.1f, -0.1f, 0);
    rightLeg();
    glPopMatrix();

    //旋转右胳膊
    glPushMatrix();
    glTranslatef(0.2f, 0.5f, 0);
    glRotatef(angx[3], 0.1f, 0, 0);
    glTranslatef(-0.2f, -0.5f, 0);
    rightArm();
    glPopMatrix();
    //旋转左胳膊
    glPushMatrix();
    glTranslatef(-0.2f, 0.5f, 0);
    glRotatef(angx[2], 0.1f, 0, 0);
    glTranslatef(0.2f, -0.5f, 0);
    leftArm();
    glPopMatrix();

    //旋转身体
    glPushMatrix();
    glTranslatef(0, 0.3f, 0);
    glRotatef(angx[1], 0, 0.1f, 0);
    glTranslatef(0, -0.3f, 0);
    Body();
    glPopMatrix();

    //旋转头
    glPushMatrix();
    glTranslatef(0, 0.6f, 0);
    glRotatef(angx[0], 0, 0.1f, 0);
    glTranslatef(0, -0.6f, 0);
    Head();
    glPopMatrix();

    glFlush(); //强制刷新缓冲
    glutSwapBuffers();//交换缓冲区域
}

int main(int argc, char *argv[]) //argc参数数量，argv参数向量。linux.
{
    //初始化窗口
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE); //设置初始显示模式,DOUBLE双缓冲
```

```
    glutInitWindowPosition(300, 0);
    glutInitWindowSize(700, 700);
    glutCreateWindow("My Robot");//调用glutMainLoop才能看到窗口

    glutKeyboardFunc(keyBoard);//先键盘
    glutDisplayFunc(&myDisplay);//myDisplay函数用来画图

    glutMainLoop();//让绘图循环进行,可以显示窗口，并且等待窗口关闭后才会返回
    return 0;
}
```