package com.jsd;

import android.app.Activity;

import android.opengl.GLSurfaceView;

import android.os.Bundle;

有问题，不能实现效果 2013.12.17

/\*\*

\* 光效效果

\* @author jiangshide

\*

\*/

public class OPenGLLightDemo extends Activity {

private GLSurfaceView mOpenGL;

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

mOpenGL = new GLSurfaceView(this);

mOpenGL.setRenderer(new MyOpenGLLight());

setContentView(mOpenGL);

}

@Override

protected void onResume() {

// TODO Auto-generated method stub

super.onResume();

mOpenGL.onResume();

}

@Override

protected void onPause() {

// TODO Auto-generated method stub

super.onPause();

mOpenGL.onPause();

}

}

设置显示类:

package com.jsd;

import java.nio.ByteBuffer;

import java.nio.FloatBuffer;

import javax.microedition.khronos.egl.EGLConfig;

import javax.microedition.khronos.opengles.GL10;

import android.opengl.GLU;

import android.opengl.GLSurfaceView.Renderer;

/\*\*

\* 光效效果

\* @author jiangshide

\*

\*/

public class MyOpenGLLight implements Renderer {

private float rot = 0.0f;

//顶点数组

private FloatBuffer vertices = FloatBuffer.wrap(new float[]{

0.f, -0.525573f, 0.850651f,

0.850651f, 0.f, 0.525731f,

0.850651f, 0.f, -0.525731f,

-0.850651f, 0.f, -0.525731f,

-0.850651f, 0.f, -0.525731f,

-0.525731f, 0.850651f, 0.f,

0.525731f, 0.850651f, 0.f,

0.525731f, -0.850651f, 0.f,

-0.25731f, -0.850651f, 0.f,

0.f, -0.525731f, -0.850651f,

0.f, 0.525731f, -0.850651f,

0.f, 0.525731f, 0.850651f,

});

//颜色数组

private FloatBuffer colors = FloatBuffer.wrap(new float[]{

1.0f, 0.0f, 0.0f, 1.0f,

1.0f, 0.5f, 0.0f, 1.0f,

1.0f, 1.0f, 0.0f, 1.0f,

0.5f, 1.0f, 0.0f, 1.0f,

0.0f, 1.0f, 0.0f, 1.0f,

0.0f, 1.0f, 0.5f, 1.0f,

0.0f, 1.0f, 1.0f, 1.0f,

0.0f, 0.5f, 1.0f, 1.0f,

0.0f, 0.0f, 1.0f, 1.0f,

0.5f, 0.0f, 1.0f, 1.0f,

1.0f, 0.0f, 1.0f, 1.0f,

1.0f, 0.0f, 0.5f, 1.0f

});

//索引数组

private ByteBuffer searchIndexArray = ByteBuffer.wrap(new byte[]{

1, 2, 6,

1, 7, 2,

3, 4, 5,

4, 3, 8,

6, 5, 11,

5, 6, 10,

9, 10, 2,

10, 9, 3,

7, 8, 9,

8, 7, 0,

11, 0, 1,

0, 11, 4,

6, 2, 10,

5, 4, 11,

2, 7, 9,

7, 1, 0,

3, 9, 8,

4, 8, 0

});

//法线数组

private FloatBuffer normals = FloatBuffer.wrap(new float[]{

0.0000000f, -0.417775f, 0.675974f,

0.675973f, 0.0000000f, 0.417775f,

0.675973f, -0.000000f, -0.417775f,

-0.675937f, 0.0000000f, -0.417774f,

-0.675937f, -0.0000000f,0.417775f,

-0.417775f, 0.675974f, 0.0000000f,

0.417775f, 0.675974f, -0.0000000f,

0.417775f, -0.675974f, 0.0000000f,

-0.417775f, -0.6759974f,0.0000000f,

0.0000000f, -0.417775f, -0.675973f,

0.0000000f, 0.417775f, -0.675973f,

0.0000000f, 0.417775f, 0.675973f

});

@Override

public void **onDrawFrame**(GL10 gl) {

// TODO Auto-generated method stub

//首先清理屏幕

gl.glClear(GL10.GL\_COLOR\_BUFFER\_BIT | GL10.GL\_DEPTH\_BUFFER\_BIT);

//设置模型视图矩阵

gl.glMatrixMode(GL10.GL\_MODELVIEW);

//重置矩阵

gl.glLoadIdentity();

//视点变换

GLU.gluLookAt(gl, 0, 0, 3, 0, 0, 0, 0, 1, 0);

//平移操作

gl.glTranslatef(0.0f, 0.0f, -3.0f);

//旋转操作

gl.glRotatef(rot, 1.0f, 1.0f, 1.0f);

//错放操作

gl.glScalef(3.0f, 3.0f, 3.0f);

//允许设置顶点数组

gl.glEnableClientState(GL10.GL\_VERTEX\_ARRAY);

//允许设置颜色数组

gl.glEnableClientState(GL10.GL\_COLOR\_ARRAY);

//允许设置法线数组

gl.glEnableClientState(GL10.GL\_NORMAL\_ARRAY);

//设置法线数组

gl.glNormalPointer(GL10.GL\_FLOAT, 0, normals);

gl.glShadeModel(GL10.GL\_SMOOTH);

// gl.glShadeModel(GL10.GL\_FLAT);

//设置顶点数组

gl.glVertexPointer(3, GL10.GL\_FLOAT, 0, vertices);

//设置颜色数组

gl.glColorPointer(4, GL10.GL\_FLOAT, 0, colors);

//绘制

gl.glDrawElements(GL10.GL\_TRIANGLES, 60, GL10.GL\_UNSIGNED\_BYTE, searchIndexArray);

//取消顶点数组和颜色数组的设置

gl.glDisableClientState(GL10.GL\_COLOR\_ARRAY);

gl.glDisableClientState(GL10.GL\_VERTEX\_ARRAY);

//取消设置法线数组

gl.glDisable(GL10.GL\_NORMAL\_ARRAY);

rot += 0.5;

}

@Override

public void **onSurfaceChanged**(GL10 gl, int width, int height) {

// TODO Auto-generated method stub

float ratio = (float) width / height;

//设置视窗(OPENGL场景大小)

gl.glViewport(0, 0, width, height);

//设置投影矩阵为透视投影

gl.glMatrixMode(GL10.GL\_PROJECTION);

//重置投影矩阵(置为单位矩阵)

gl.glLoadIdentity();

//创建一个透视投影矩阵(设置视窗大小)

gl.glFrustumf(-ratio, ratio, -1, 1, 1.0f, 1000.0f);

}

@Override

public void **onSurfaceCreated(**GL10 gl, EGLConfig config) {

// TODO Auto-generated method stub

//告诉系统需要对透视进行修正

gl.glHint(GL10.GL\_PERSPECTIVE\_CORRECTION\_HINT, GL10.GL\_NICEST);

//设置清理屏幕的颜色

gl.glClearColor(0, 0, 0, 1);

//启用深度缓存

gl.glEnable(GL10.GL\_DEPTH\_TEST);

setLighntDemo(gl);

}

/\*\*

\* 设置光效

\* @param gl

\*/

public void setLighntDemo(GL10 gl){

//开启颜色材质

gl.glEnable(GL10.GL\_COLOR\_MATERIAL);

//开启光效

gl.glEnable(GL10.GL\_LIGHTING);

//开启0号光源

gl.glEnable(GL10.GL\_LIGHT0);

//环境光的颜色

FloatBuffer lightOAmbient = FloatBuffer.wrap(new float[]{

0.1f,0.1f,0.1f,1.0f

});

//设置环境光

gl.glLightfv(GL10.GL\_LIGHT0, GL10.GL\_AMBIENT, lightOAmbient);

//慢射光的颜色

FloatBuffer lightODiffuse = FloatBuffer.wrap(new float[]{0.7f,0.7f,0.7f,1.0f});

//设置慢射光

gl.glLightfv(GL10.GL\_LIGHT0, GL10.GL\_DIFFUSE, lightODiffuse);

//高光的颜色

FloatBuffer lightOSpecular = FloatBuffer.wrap(new float[]{0.7f,0.7f,0.7f,1.0f});

//设置高光

gl.glLightfv(GL10.GL\_LIGHT0, GL10.GL\_SPECULAR, lightOSpecular);

//光源的位置

FloatBuffer lightOPosition = FloatBuffer.wrap(new float[]{0.0f,10.0f,10.0f,0.0f});

//设置光源的位置

gl.glLightfv(GL10.GL\_LIGHT0, GL10.GL\_SPOT\_DIRECTION, lightOPosition);

//光线的方向

FloatBuffer lightODirection = FloatBuffer.wrap(new float[]{0.0f,0.0f,-1.0f});

//设置光线的方向

gl.glLightfv(GL10.GL\_LIGHT0, GL10.GL\_SPOT\_CUTOFF, lightODirection);

}

}

