可以显示效果

public class MainActivity extends Activity {

GLSurfaceView surfaceView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

//创建GLsurfaceView

surfaceView = new GLSurfaceView(this);

//设置渲染器

surfaceView.setRenderer(new GalleryRenderer(this));

LinearLayout main = (LinearLayout)findViewById(R.id.main);

main.addView(surfaceView);

}

}

public class GalleryRenderer implements Renderer {

private Square square = new Square();

private Context mContext;

//纹理Id

private int textureId;

//构造函数

public GalleryRenderer(Context context) {

mContext = context;

init();

}

//初始化工作

private void init(){

}

@Override

public void onDrawFrame(GL10 gl) {

// 清除屏幕和深度缓存

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

gl.glLoadIdentity();

gl.glTranslatef(0, 0, -4);

// 绘制正方形

square.draw(gl);

}

@Override

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

// 设置画面的大小

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

// 设置投影矩阵

gl.glMatrixMode(GL10.GL\_PROJECTION);

// 重置投影矩阵

gl.glLoadIdentity();

// 设置画面比例

GLU.gluPerspective(gl, 45.0f, (float) width / (float) height, 0.1f,100.0f);

// 选择模型观察矩阵

gl.glMatrixMode(GL10.GL\_MODELVIEW);

// 重置模型观察矩阵

gl.glLoadIdentity();

}

@Override

public void onSurfaceCreated(GL10 gl, EGLConfig arg1) {

// 黑色背景

gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f);

// 启用阴影平滑（不是必须的）

gl.glShadeModel(GL10.GL\_SMOOTH);

// 设置深度缓存

gl.glClearDepthf(1.0f);

// 启用深度测试

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

// 所作深度测试的类型

gl.glDepthFunc(GL10.GL\_LEQUAL);

// 对透视进行修正

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

//启动纹理

gl.glEnable(GL10.GL\_TEXTURE\_2D);

loadTexture(gl);

}

/\*\*

\* 加载纹理图片

\* @param gl

\*/

private void loadTexture(GL10 gl) {

InputStream bitmapStream = null;

Bitmap bitmap = null;

try {

// 打开图片资源流

bitmapStream = mContext.getResources().openRawResource(

R.drawable.ic\_launcher);

// 解码图片生成 Bitmap 实例

bitmap = BitmapFactory.decodeStream(bitmapStream);

// 生成一个纹理对象，并将其ID保存到成员变量 texture 中

int[] textures = new int[1];

gl.glGenTextures(1, textures, 0);

textureId = textures[0];

// 将生成的空纹理绑定到当前2D纹理通道

gl.glBindTexture(GL10.GL\_TEXTURE\_2D, textureId);

// 设置2D纹理通道当前绑定的纹理的属性

gl.glTexParameterf(GL10.GL\_TEXTURE\_2D, GL10.GL\_TEXTURE\_MIN\_FILTER,

GL10.GL\_NEAREST);

gl.glTexParameterf(GL10.GL\_TEXTURE\_2D, GL10.GL\_TEXTURE\_MAG\_FILTER,

GL10.GL\_LINEAR);

gl.glTexParameterf(GL10.GL\_TEXTURE\_2D, GL10.GL\_TEXTURE\_WRAP\_S,

GL10.GL\_REPEAT);

gl.glTexParameterf(GL10.GL\_TEXTURE\_2D, GL10.GL\_TEXTURE\_WRAP\_T,

GL10.GL\_REPEAT);

// 将bitmap应用到2D纹理通道当前绑定的纹理中

GLUtils.texImage2D(GL10.GL\_TEXTURE\_2D, 0, bitmap, 0);

} finally {

// 释放资源

// BTW: 期待 android 早日支持 Java 新的 try-with-resource 语法

if (bitmap != null)

bitmap.recycle();

if (bitmapStream != null) {

try {

bitmapStream.close();

} catch (IOException e) {

}

}

}

}

}

public class Square {

// 顶点坐标数组

private float vertices[] = { -1.0f, 1.0f, 0.0f, // 0, 左上

-1.0f, -1.0f, 0.0f, // 1, 左下

1.0f, -1.0f, 0.0f, // 2, 右下

1.0f, 1.0f, 0.0f, // 3, 右上

};

float texCoords[] = new float[] {

// FRONT

0.0f, 0.0f,

1.0f, 0.0f,

0.0f, 1.0f,

1.0f, 1.0f,

};

// 连接规则

private short[] indices = { 0, 1, 2, 0, 2, 3 };

// 顶点缓存

private FloatBuffer vertexBuffer;

// 索引缓存

private ShortBuffer indexBuffer;

//纹理缓冲

private FloatBuffer textureBuffer;

public Square() {

// 一个float为4 bytes, 因此要乘以4

ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length \* 4);

vbb.order(ByteOrder.nativeOrder());

vertexBuffer = vbb.asFloatBuffer();

vertexBuffer.put(vertices);

vertexBuffer.position(0);

// short类型同理

ByteBuffer ibb = ByteBuffer.allocateDirect(indices.length \* 2);

ibb.order(ByteOrder.nativeOrder());

indexBuffer = ibb.asShortBuffer();

indexBuffer.put(indices);

indexBuffer.position(0);

ByteBuffer textbb = ByteBuffer.allocateDirect(texCoords.length \* 4);

textbb.order(ByteOrder.nativeOrder());

textureBuffer = textbb.asFloatBuffer();

textureBuffer.put(texCoords);

textureBuffer.position(0);

}

/\*\*

\* 绘制正方形到屏幕

\*

\* @param gl

\*/

public void draw(GL10 gl) {

// 逆时针环绕

gl.glFrontFace(GL10.GL\_CCW);

// 开启剔除功能

gl.glEnable(GL10.GL\_CULL\_FACE);

// 剔除背面

gl.glCullFace(GL10.GL\_BACK);

// 开启顶点缓存写入功能

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

gl.glEnableClientState(GL10.GL\_TEXTURE\_COORD\_ARRAY);

// 设置顶点

// size:每个顶点有几个数指描述。

// type:数组中每个顶点的坐标类型。

// stride:数组中每个顶点间的间隔，步长（字节位移）。

// pointer:存储着每个顶点的坐标值。初始值为0

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

gl.glTexCoordPointer(2, GL10.GL\_FLOAT, 0, textureBuffer);

// 绑定纹理

//gl.glBindTexture(GL10.GL\_TEXTURE\_2D, texture);

gl.glDrawElements(GL10.GL\_TRIANGLES, indices.length,

GL10.GL\_UNSIGNED\_SHORT, indexBuffer);

// 关闭各个功能

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

gl.glDisableClientState(GL10.GL\_TEXTURE\_COORD\_ARRAY);

gl.glDisable(GL10.GL\_CULL\_FACE);

}

}