test2.java

**package** my.test2;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.opengl.\*;

**public class** test2 **extends** Activity {

    /\*\* Calledwhen the activity is first created. \*/

    GLSurfaceView myRenderer;

    @Override

**public void onCreate**(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

        myRenderer=**new** GLSurfaceView(**this**);

        myRenderer.setRenderer(**new** Renderer());

        setContentView(myRenderer);

    }

}

Renderer.java

**package** my.test2;

**import** java.nio.ByteBuffer;

**import** java.nio.ByteOrder;

**import** java.nio.FloatBuffer;

**import** java.nio.ShortBuffer;

**import** javax.microedition.khronos.egl.EGLConfig;

**import** javax.microedition.khronos.opengles.GL10;

**import** android.opengl.GLSurfaceView;

**public class** Renderer **implements**GLSurfaceView.Renderer{

    ////////////定义要使用的颜色/////////////

    ////////////定义缓冲区////////////////////

**private** FloatBuffer vertexBuffer;

    ////////////定义角度/////////////////////

    ////////////定义矩阵////////////////////

**private float**[] vertexArray={

           -0.5f,0.5f,0,

           0.5f,0.5f,0,

           0.5f,-0.5f,0,

           -0.5f,-0.5f,0

    };

    @Override

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

       // **TODO** Auto-generated method stub

       gl.glClearColor(0f,0f,0f,1.0f);

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

       gl.glColor4f(0.5f,0.5f, 0.5f, 0.5f);

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

       gl.glDrawArrays(GL10.*GL\_TRIANGLE\_FAN*, 0, 4);

       //gl.glDrawElements(GL10.GL\_TRIANGLES,3, GL10.GL\_UNSIGNED\_SHORT, indexBuffer);

    }

    @Override

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

       // **TODO** Auto-generated method stub

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

    }

    @Override

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

       // **TODO** Auto-generated method stub

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

       init();

    }

**private void init**(){

       ByteBuffer vbb=ByteBuffer.*allocateDirect*(4\*3\*4);

       vbb.order(ByteOrder.*nativeOrder*());

       vertexBuffer=vbb.asFloatBuffer();

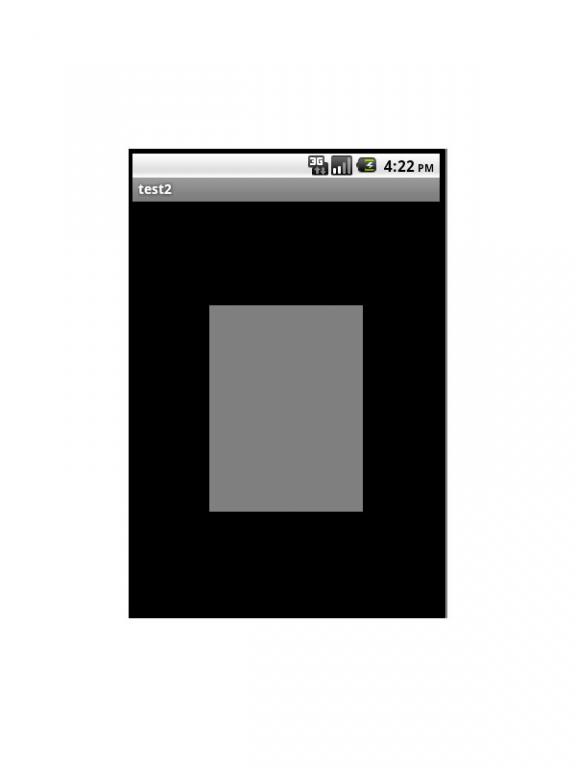
       vertexBuffer.put(vertexArray);

       vertexBuffer.position(0);

    }

}

效果图：



glDrawArray它有多种绘图模式

**GL\_POINTS  
GL\_LINES  
GL\_LINE\_LOOP  
GL\_LINE\_STRIP  
GL\_TRIANGLES  
GL\_TRIANGLE\_STRIP  
GL\_TRIANGLE\_FAN**

下面主要讲解一下后3个模式

这三个模式都是连续绘制三角形的，三角形的个数由模式类型，以及顶点矩阵中的顶点个数决定的。

GL\_TRIANGLES

这个参数意味着OpenGL使用三个顶点来组成图形。所以，在开始的三个顶点，将用顶点1，顶点2，顶点3来组成一个三角形。完成后，在用下一组的三个顶点(顶点4，5，6)来组成三角形，直到数组结束。

**GL\_TRIANGLE\_STRIP**

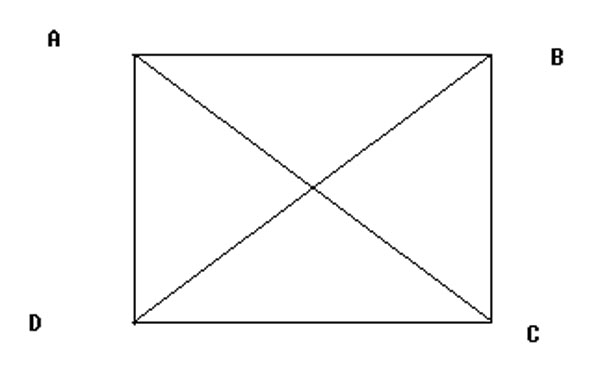
OpenGL的使用将最开始的两个顶点出发，然后遍历每个顶点，这些顶点将使用前2个顶点一起组成一个三角形。

**GL\_TRIANGLE\_FAN**

在跳过开始的2个顶点，然后遍历每个顶点，让OpenGL将这些顶点于它们前一个，以及数组的第一个顶点一起组成一个三角形。

在前面的代码中，我们用的模式是**GL\_TRIANGLE\_FAN**，看下图

效果图：



使用**GL\_TRIANGLE\_FAN，**绘出三角形的顺序是三角形ABC，然后是三角形ACD。

若使用**GL\_TRIANGLE\_STRIP，**则效果图为：



绘出三角形的顺序是三角形ABC，然后是三角形BCD。