Ⅰ按正常流程新建一个 weex 程序

```
npm
PS D:\AndroidProject> weex create triangle

     Update available 1.2.0 → 1.2.3
     Run weex update weexpack@1.2.3 to update

? Project name triangle
? Project description A weex project
? Author xu <iamxuting@foxmail.com>
? Select weex web render latest
? Babel compiler (https://babeljs.io/docs/plugins/#stage-x-experimental-presets) stage-0
? Use vue-router to manage your view router? (not recommended) Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Standard
? Set up unit tests Yes
? Should we run `npm install` for you after the project has been created? (recommended) npm
```

Ⅱ 加 Android 模块

$weex platform add android

Ⅲ 在 platforms\android\app\src\main\java\com\weex\app\extend 目录下添加 GLSurfaceView（组件的样式，动作，此文件和 native 写法相同如有代码，直接将文件复制过来改一下包名即可）

```
package com.weex.app.extend;

import android.opengl.GLSurfaceView;
import android.opengl.GLU;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

/**
 * Created by Administrator on 2016/9/26.
 * <p>
 * 定义一个统一图形绘制的接口
 */

public class OpenGLRenderer implements GLSurfaceView.Renderer {
    /**
     * 主要用来设置一些绘制时不常变化的参数，例如：背景色，是否打开 Z-buffer(去除隐藏面)等
     *
     * @param gl
     * @param config
```

```
     */
     //添加成员 privateTriangle mTriangle 并在构造函数中初始化(调用
Triangle.java)
     private Triangle mTriangle;
     public OpenGLRenderer()
     {
       mTriangle = new Triangle();
      }
    @Override
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        //设置背景的颜色
        gl.glClearColor(0f, 1f, 0f, 0.5f);
        //使光滑的材质,默认不需要。
        gl.glShadeModel(GL10.GL_SMOOTH);
        //深度缓冲设置。
        gl.glClearDepthf(1.0f);
        //启用深度测试。
        gl.glEnable(GL10.GL_DEPTH_TEST);
        //深度测试类型
        gl.glDepthFunc(GL10.GL_LEQUAL);
        //最好的的角度计算。
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_NICEST);
        //启用顶点数组
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
    }

    /**
     * 如果设备支持屏幕的横向和纵向切换,
     * 这个方法将发生在横向<==>纵向互换时,
     * 此时可以重新设置绘制的纵横比率。
     *
     * @param gl
     * @param width
     * @param height
     */
    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height) {

        //将当前视图端口设置为新的大小。
        gl.glViewport(0, 0, width, height);
        //选择投影矩阵
        gl.glMatrixMode(GL10.GL_PROJECTION);
        //重置投影矩阵
        gl.glLoadIdentity();
```

```
        /**
         * 建立一个透视投影矩阵
         *
         *  gl : GL10 接口
         * fovy : 指定领域的视角,在 Y 轴方向。指定方面定量 determin 领域在 x 方向上的看法。
         *        高宽比的比例是 x(宽度)y(高度)。
         * zNear : 指定观众的距离不远的剪裁平面的(总是正数)。
         * zFar : 指定了与观众的距离遥远的剪裁平面的(总是正数)。
         */
        //计算窗口的长宽比
        GLU.gluPerspective(gl, 45.0f, (float) width / (float) height,
0.1f, 100.0f);
        //选择 modelview 矩阵
        gl.glMatrixMode(GL10.GL_MODELVIEW);
        //重置投影矩阵
        gl.glLoadIdentity();

    }

    /**
     * 定义实际的绘图操作
     *
     * @param gl
     */
    @Override
    public void onDrawFrame(GL10 gl) {
        //清除屏幕和深度缓冲。
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
        // 设置当前矩阵为模型视图模式
        gl.glMatrixMode(GL10.GL_MODELVIEW);
        gl.glLoadIdentity();  // reset the matrix to its default state
        // 设置视点
        GLU.gluLookAt(gl, 0, 0, -5, 0f, 0f, 0f, 0f, 1.0f, 0.0f);
        mTriangle.draw(gl);
    }
}
```

Ⅳ 添加三角形绘制文件 Triangle.java(此文件和 native 写法相同如有代码,直接将文件复制过来改一下包名即可)

```
package com.weex.app.extend;
```

```
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;

import javax.microedition.khronos.opengles.GL10;
public class Triangle {
  public Triangle()
  {
     float triangleCoords[] = {
         // X, Y, Z 这是一个等边三角形
         -0.5f, -0.25f, 0,
          0.5f, -0.25f, 0,
          0.0f, 0.559016994f, 0
        };
        // 初始化三角形的顶点缓存
        ByteBuffer vbb = ByteBuffer.allocateDirect(
            // (# of coordinate values * 4 bytes per float)
            triangleCoords.length * 4);
        vbb.order(ByteOrder.nativeOrder());// 使用设备硬件本身的字节序
        triangleVB = vbb.asFloatBuffer(); // 从 ByteBuffer 中创建一个浮点缓
存
        triangleVB.put(triangleCoords); // 向浮点缓存中添加顶点坐标
        triangleVB.position(0); // 使缓存读第一个坐标
  }
  public void draw(GL10 gl)
  {
    gl.glColor4f(0.63671875f, 0.76953125f, 0.22265625f, 0.0f); //设置当前
颜色
    gl.glVertexPointer(3, GL10.GL_FLOAT, 0, triangleVB);//设置顶点
    gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);//绘制三角形
  }
  private FloatBuffer triangleVB;
}
```

Ⅴ 添加 native 组件源代码封装成 WXComponent 文件 glsurface.java

```
package com.weex.app.extend;
//下面的 import 无论什么组件都要有
import android.content.Context;
import android.support.annotation.NonNull;
//下面的 import 根据原生 natvie 组件的不同而不同，一般根据 Android studio 中界
面 activity 中的 import 引用
import android.opengl.GLSurfaceView;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```

```
import android.view.Window;
import android.view.WindowManager;

//下面的 import 无论什么组件都要有
import com.taobao.weex.WXSDKInstance;
import com.taobao.weex.dom.WXDomObject;
import com.taobao.weex.ui.component.WXComponent;
import com.taobao.weex.ui.component.WXComponentProp;
import com.taobao.weex.ui.component.WXVContainer;

public class glsurface extends WXComponent<GLSurfaceView> {
//以下语句块无论什么组件都要有
  public glsurface(WXSDKInstance instance, WXDomObject dom, WXVContainer
parent) {
    super(instance, dom, parent);
  }

  @Override
  protected GLSurfaceView initComponentHostView(@NonNull Context
context) {
    GLSurfaceView view = new GLSurfaceView(context);
    //setRenderer 方法用于把渲染器注册到 GLSurfaceView
    view.setRenderer(new OpenGLRenderer());
    return view;
  }

}
```

Ⅵ 在 WXApplication.java 中添加 import 和注册自定义组件 glsurface

```
import com.weex.app.extend.glsurface;
import com.weex.app.extend.OpenGLRenderer;
import com.weex.app.extend.Triangle;
```

```
WXSDKEngine.registerComponent("glsurface", glsurface.class);
```

Ⅶ 在 weex 项目 src 目录下的.vue 文件应用自定义组件

```
<!--index.vue-->
<template>
  <div>
    <text class="text">GLSurfaceView is working!</text>
    <glsurface class="gls"></glsurface>
  </div>
</template>
<style>
```

```
.text{
  font-size: 30px;
  color: blueviolet;
}
.gls {
  width:800px;
  height:1600px;
}
</style>
```

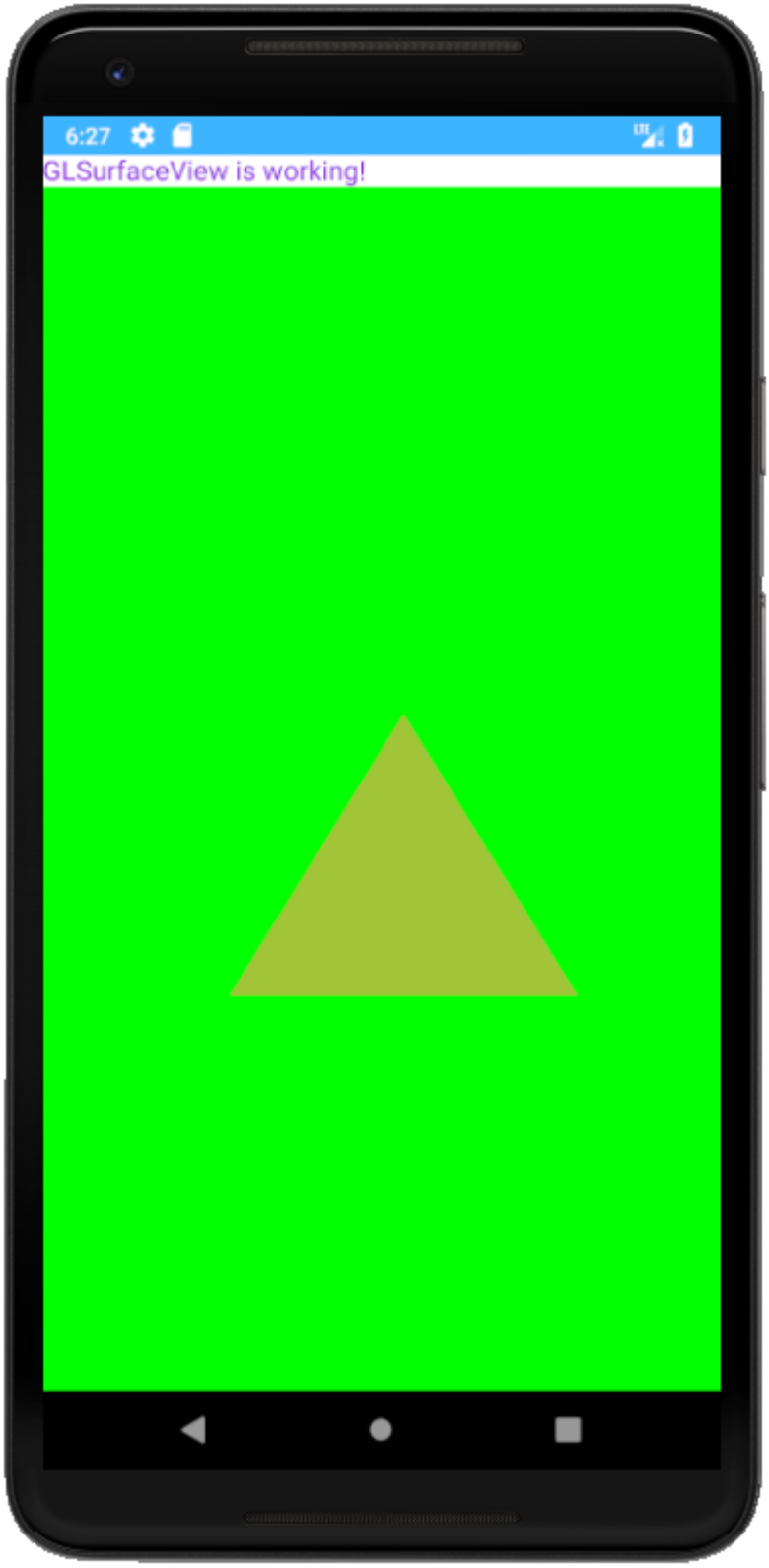
Ⅷ 直接调试运行或打包 Android

$weex run android

或

$weex build android

运行效果如下


6:27
GLSurfaceView is working!