来自 https://blog.csdn.net/ouruolaxuhui/article/details/51062255

OpenGL 即 Open Graphics Library(开放的图形库接口)，主要用于三维图形编程。

OpenGL ES：OpenGL 的子集，嵌入式开放图形库接口。OpenGL ES 提供了 GLSurfaceView 组件。

GLSurfaceView 用于显示 3D 图像，本身并不提供绘制 3D 图形的功能，绘制功能由 GLSurfaceView.Renderer 来完成。

使用 OpenGL ES 的步骤如下：

1.创建 GLSurfaceView 组件（也可继承该组件），并使用 Activity 显示 GLSurfaceView 组件；

```
1.  GLSurfaceView glSurfaceView=new GLSurfaceView(this);
```

2、为 GLSurfaceView 创建 GLSurfaceView.Renderer 实例，并实现 GLSurfaceView.Renderer 接口的三个方法：
其中，GL10 代表 OpenGL 的绘制画笔

```
1.  MyRenderer implements Renderer{
2.  //onSurfaceCreated 方法主要用于执行一些初始化操作
3.      @Override
4.      public void onSurfaceCreated(GL10 gl, EGLConfig config) {
5.          gl.glDisable(GL10.GL_DITHER);// gl.glDisable 用于禁用 OpenGL 某方面的特性，该处表示关闭抗抖动，可以提高性能
6.          gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_FASTEST);//该方法用于修正，本处用于设置对透视进行修正
7.          gl.glClearColor(0, 0, 0, 0);//设置 OpenGL 清屏所用的颜色，四个参数分别代表红、绿、蓝和透明度值，范围为 0-1，此处表示黑色
8.          gl.glEnable(GL10.GL_DEPTH_TEST);//启用某方面的性能，此处为启动深度测试，负责跟踪每个物体在 Z 轴上的深度，避免后面的物体遮挡前面的物体
9.          gl.glDepthFunc(GL10.GL_LEQUAL);//设置深度测试的类型，此处为如果输入的深度值小于或等于参考值，则通过
10.         gl.glShadeModel(GL10.GL_SMOOTH);//设置阴影模式为平滑模式
11. }
12.
13.   //当 SurfaceView 大小改变时回调，通常用于初始化 3D 场景
14.   @Override
15.   public void onSurfaceChanged(GL10 gl, int width, int height) {
16.         gl.glViewport(0, 0, width, height);//设置 3D 视窗的位置与大小
17.         gl.glMatrixMode(GL10.GL_PROJECTION);//设置矩阵模式为投影矩阵，这意味着越远的东西看起来越小，GL10.GL_MODELVIEW：模型视图矩阵：任何新的变换都会影响          该矩阵中的所有物体
18.         float ratio=(float)width/height;
```

```
19.         gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10);//设置透视投影的空间大小，前两个参数用于设置 X 轴的最小值与最大值，中间两个参数用于设置 y 轴的最小值最大值            ，后两个参数用于设置 Z 轴的最小值最大值
20.   }
21.   //用于绘制 3D 图形
22.   @Override
23.   public void onDrawFrame(GL10 gl) {
24.         //清楚屏幕缓存和深度缓存(一般为必须设置的)
25.         gl.glClear(GL10.GL_COLOR_BUFFER_BIT|GL10.GL_DEPTH_BUFFER_BIT);
26.         gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);//启用顶点坐标数组
27.         gl.glEnableClientState(GL10.GL_COLOR_ARRAY);//启用顶点颜色数组
28.         gl.glMatrixMode(GL10.GL_MODELVIEW);设置矩阵模式为模型视图矩阵
29.         gl.glLoadIdentity();//相当于 reset()方法，用于初始化单位矩阵
30.         gl.glTranslatef(-0.32f, 0.35f, -1.1f);//移动绘图中心
```

```
<pre code_snippet_id="1634794" snippet_file_name="blog_20160405_2_9523416" name="code" class="java"><span style="font-family: Arial, Helvetica, sans-serif;">
gl.glVertexPointer(3, GL10.GL_FLOAT, 0, triangleFloatBuffer);//设置顶点的位置数据，第一个参数指定多少个元素确定一个顶点，通常为 3；第二个参数指定顶点坐标值的 类型，triangleFloatBuffer 是一维数组，形如，(x1,y1,z1,x2,y2,z2,...)，用于指定顶点坐标值
</span>
```

```
1.  gl.glColorPointer(4, GL10.GL_FIXED, 0, triangleColorBuffer);//设置顶点的颜色数据
```

```
1.  gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);//根据顶点绘制平面图形，第一个参数表示绘制图形的类型，GL10.GL_TRIANGLES：绘制简单的三角形，GL10.GL_TRIANG            LE_STRIP:沿着给出的顶点数据绘制三角形来形成平面图形。OpenGL 只能绘制三角形组成 3D 图形；第二个参数指定从哪个顶点开始，第三个参数表示顶点的数量
2.          gl.glFinish();//绘制结束
3.          gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
4.    }
5.  }
```

注：输入顶点数组时，数组中每个顶点的顺序不同，可能会导致画出的图形不一样

3、使用 setRenderer 方法为 GLSurfaceView 添加 Renderer。

示例：使用 GLSurfaceView 绘制三角形

http://www.jb51.net/article/96080.htm

定义三角形

OpenGL 允许我们使用三维坐标来定义物体。在绘制三角形前，我们需要定义它各个点的坐标。我们一般使用数组来存储各个顶点的坐标。

OpenGL ES 默认 [0,0,0] (X,Y,Z) 在 GLSurfaceView 的中心，[1,1,0]在右上角，[-1,-1,0]在左下角。

绘制三角形

在绘制三角形之前，我们必须告诉 OpenGL 我们正在使用顶点数组。然后我们才使用绘制函数画出三角形。

实验步骤：

1. 添加新的类 Triangle

代码如下：

```
1  public class Triangle {
2      public Triangle()
3      {
4            float triangleCoords[] = {
5                      // X, Y, Z 这是一个等边三角形
6                      -0.5f, -0.25f, 0,
7                        0.5f, -0.25f, 0,
8                        0.0f, 0.559016994f, 0
9                  };
10                 // 初始化三角形的顶点缓存
11                 ByteBuffer vbb = ByteBuffer.allocateDirect(
12                         // (# of coordinate values * 4 bytes per float)
13                         triangleCoords.length * 4);
14                 vbb.order(ByteOrder.nativeOrder());// 使用设备硬件本身的字节序
15                 triangleVB = vbb.asFloatBuffer(); // 从 ByteBuffer 中创建一个浮点缓存
16                 triangleVB.put(triangleCoords); // 向浮点缓存中添加顶点坐标
17                 triangleVB.position(0); // 使缓存读第一个坐标
18     }
19     public void draw(GL10 gl)
20     {
21         gl.glColor4f(0.63671875f, 0.76953125f, 0.22265625f, 0.0f); //设置当前颜色
22         gl.glVertexPointer(3, GL10.GL_FLOAT, 0, triangleVB);//设置顶点
23         gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);//绘制三角形
24     }
25     private FloatBuffer triangleVB;
```

```
18  }
```

2. 在 myGLRenderer 类中添加成员 privateTriangle mTriangle 并在构造函数中初始化。

代码如下：

```
1  public myGLRenderer()
2  {
3          mTriangle = new Triangle();
   }
```

3. 在 myGLRenderer 类的 onSurfaceCreated()函数最后添加 glEnableClientState()方法来启用顶点数组。

代码如下：

```
1
2  @Override
   public void onSurfaceCreated(GL10 gl, EGLConfig config) {
3          // TODO Auto-generated method stub
4          gl.glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
           gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
5  }
6
```

4. 在 myGLRenderer 类的 onDrawFrame()函数最后添加三角形绘制方法。

代码如下：

```
1
2  @Override
   public void onDrawFrame(GL10 gl) {
3          // TODO Auto-generated method stub
4          gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
           mTriangle.draw(gl);
5  }
6
```

这样，我们便完成了一个平面三角形的绘制。但我们发现这个三角形并不像我们定义的那样是一个等边三角形，这是由于 OpenGL 总假设我们的屏幕是一个正方形，这样在绘制的时候最终图形会随着屏幕长宽比例的不同而被拉伸。为了得到正确的显示，我们需要将图形投影到正确的位置。这一功能我们在下一节进行实现。

Android 设备屏幕通常不是正方形的，而 OpenGL 总是默认地将正方形坐标系投影到这一设备上，这就导致图形无法按真实比例显示。要解决这一问题，我们可以使用 OpenGL 的投影模式和相机视图将图形的坐标进行转换以适应不同的设备显示。

实验步骤：

1. 修改 myGLRenderer 类的 onSurfaceCreated()函数来启用 GL10.GL_PROJECTION 模式，计算屏幕的长宽比并使用这一比例来转换物体的坐标。

代码如下：

```
1
2  @Override
3  public void onSurfaceChanged(GL10 gl, int width, int height) {
4          // TODO Auto-generated method stub
5          gl.glViewport(0, 0, width, height);
6          float ratio = (float) width / height;
7          gl.glMatrixMode(GL10.GL_PROJECTION); // 设置当前矩阵为投影矩阵
8          gl.glLoadIdentity(); // 重置矩阵为初始值
9          gl.glFrustumf(-ratio, ratio, -1, 1, 3, 7); // 根据长宽比设置投影矩阵
   }
```

2. 修改 myGLRenderer 的 onDrawFrame()方法，启用 MODELVIEW 模式，并使用 GLU.gluLookAt()来设置视点。

代码如下：

```
1
2  @Override
3  public void onDrawFrame(GL10 gl) {
4      // TODO Auto-generated method stub
5          gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
6          // 设置当前矩阵为模型视图模式
7          gl.glMatrixMode(GL10.GL_MODELVIEW);
8          gl.glLoadIdentity();   // reset the matrix to its default state
9          // 设置视点
10         GLU.gluLookAt(gl, 0, 0, -5, 0f, 0f, 0f, 0f, 1.0f, 0.0f);
11         mTriangle.draw(gl);
   }
```

这样，我们绘制的图形比例就总是正确的，不再受设备的影响而被拉伸变形了。