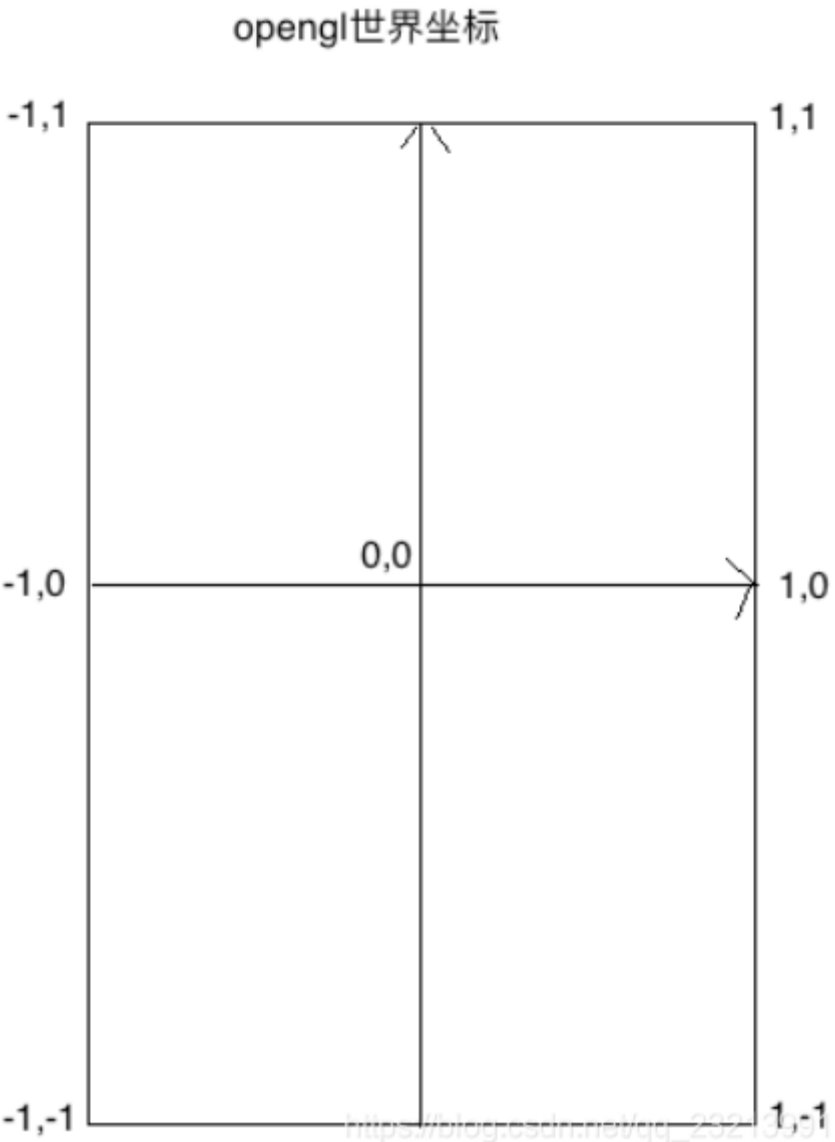
什么是OpenGL?

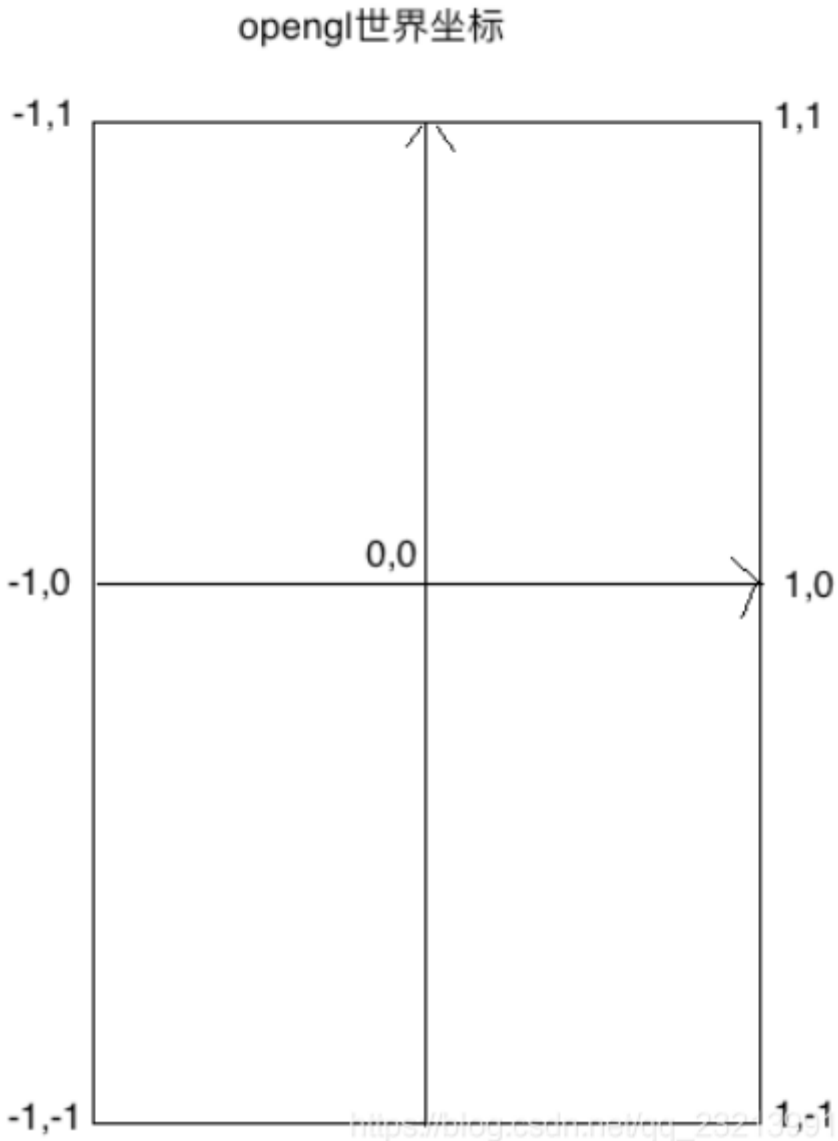
openGL是一个图形绘制专业编程接口，功能比较强大，可以绘制二维，三维，它与硬件没有关系，也可以在不同的平台上使用，

进行良好的移植，使用较为广泛

> 分析OpenGL坐标系和android坐标系



> android坐标系



android的坐标点要用opengl的方式来显示就要换算成opengl的方式，就像是2长图片的重合坐标上的修改，下面直接撸代码

始之前，我们需要建立2个文件，

> 一个是顶点着色器，
>
> 一个是片源着色器

顶点着色器是处理顶点、法线等数据，片元着色器是处理光、阴影、遮挡、环境等等对物体表面的影响，最终生成一副图像

在res下面建立一个文件，camera.vert顶点 camera_planyuan.frag片元

下面介绍一下数据类型：float　浮点型

```
vec2    含两个浮点型数据的向量

vec4   含四个浮点型数据的向量(xyzw,rgba,stpq)

sampler2D   2D纹理采样器(代表一层纹理)

内置函数exture2D (采样器,坐标)   采样指定位置的纹理

内置变量 顶点    gl_Position       vec4 顶点位置

片元   gl_FragColor    vec4 颜色
```

需求

使用OpenGL显示摄像头，分析一下需求，其实也就是两步，第一步采集摄像头数据，第二步将摄像头数据显示到屏幕上

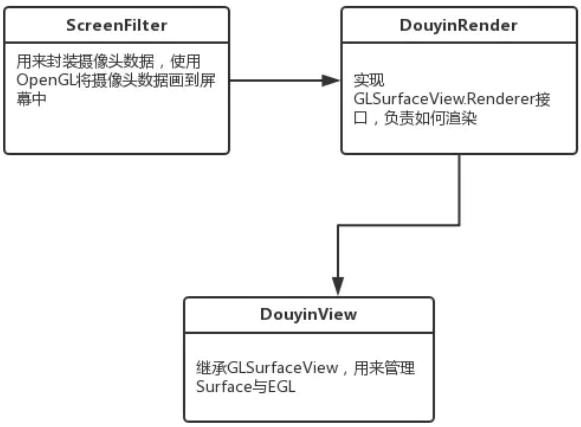
采集摄像头数据

使用Android的Camera就可以实现采集

将摄像头数据显示到屏幕上

这里显示到屏幕上，其实有很多方法，比如ANative_window等等，但是这个是使用OpenGL实现，使用OpenGL怎么实现呢，在上一篇博客中说到了OpenGL的绘制流程，基本就是按照那个流程进行实现的。

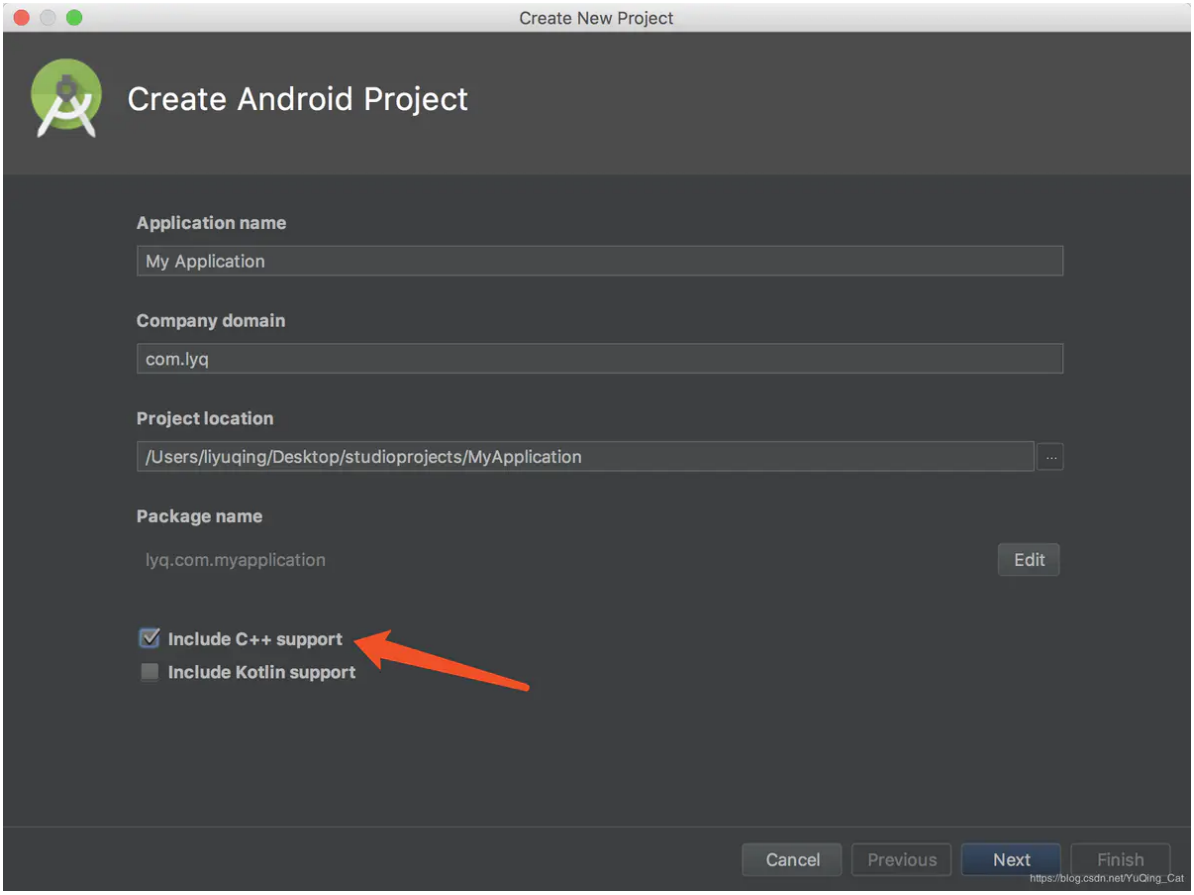
下面是项目结构中一个不太规范的类图，下面根据这种图来实现具体的代码



简单点说，我们都知道SurfaceView实质是将底层显存Surface显示到界面上，而GLSurfaceView实际就是在这个基础之上增加了OpenGL环境，DouyinView实际就相当于一块画布，而ScreenFilter中是封装了如何使用画笔去画当前摄像头采集到的内容，而DouyinRender渲染器，就是将ScreenFilter中的画渲染到画布上。

创建工程

创建一个JNI工程，就是在一开始创建项目的时候，勾选上 Include C++ support,这样就创建好了



配置文件 AndroidManifest.xml

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="lyq.com.douyin">

    <!--表示只有在支持OpenGL ES 2.0版本手机上可运行-->
    <uses-feature android:glEsVersion="0x00020000" android:required="true"/>
    <uses-permission android:name="android.permission.CAMERA"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="DouYin"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```


OpenGL的使用还需要有设备制造商提供支持，以下是设备的支持情况，项目里都是用来2.0

OpenGL ES 1.0 和 1.1 ： Android 1.0和更高的版本支持这个API规范。

OpenGL ES 2.0 ： Android 2.2(API 8)和更高的版本支持这个API规范。

OpenGL ES 3.0 ： Android 4.3(API 18)和更高的版本支持这个API规范。

OpenGL ES 3.1 : Android 5.0(API 21)和更高的版本支持这个API规范。

DouyinView.java

```java
public class DouyinView extends GLSurfaceView {


    public DouyinView(Context context) {
        this(context, attrs: null);
    }

    public DouyinView(Context context, AttributeSet attrs) {
        super(context, attrs);

        //1.配置EGL版本(必须设置这个)
        setEGLContextClientVersion(2);
        //2.设置渲染器 (后面会着重说这个类)
        setRenderer(new DouyinRender( mView: this));
        //设置渲染模式
        /**
         * 有两种模式
         * 1.连续渲染
         * 2.按需渲染
         * 这里选择的是按需渲染
         */
        setRenderMode(RENDERMODE_WHEN_DIRTY);

    }
}
```


DouyinRender.java

```java
public class DouyinRender implements GLSurfaceView.Renderer, SurfaceTexture.OnFrameAvailableListener {

    private ScreenFilter mScreenFilter;
    private DouyinView mView;
    private CameraHelper mCameraHelper;
    private SurfaceTexture mSurfaceTexture;
    private int[] mTextures;
    private float[] mtx = new float[16];

    public DouyinRender(DouyinView mView) {
        this.mView = mView;
    }

    /**
     * 画布创建好了
     * @param gl
     * @param config
     */
    @Override
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {...}

    /**
     * 画布改变了
     * @param gl
     * @param width
     * @param height
     */
    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height) {...}


    /**
     * 开始画画了
     * @param gl
     */
    @Override
    public void onDrawFrame(GL10 gl) {...}

    /**
     * 当有一个可用帧时调用
     * @param surfaceTexture
     */
    @Override
    public void onFrameAvailable(SurfaceTexture surfaceTexture) { mView.requestRender(); }
}
```


这里使用的Render,是实现了OpenGL配置好EGL的渲染器，后面会自己来配置，这里先使用这个，上面也有提到过，OpenGL的操作都是在GLThread线程中完成，所以这里的onSurfaceCreated，onSurfaceChanged，onDrawFrame都是在GLThread中实现的，下面单独看看每个方法的使用

```
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
    //初始化操作
    mCameraHelper = new CameraHelper(Camera.CameraInfo.CAMERA_FACING_BACK);

    //准备好摄像头绘制的画布
    //通过gl创建一个纹理id
    mTextures = new int[1];
    GLES20.glGenTextures(mTextures.length,mTextures, offset: 0);
    mSurfaceTexture=new SurfaceTexture(mTextures[0]);
    mSurfaceTexture.setOnFrameAvailableListener(this);

    //必须要glThread中进行初始化
    mScreenFilter=new ScreenFilter(mView.getContext());

}
```


CameraHelper是封装了对Camera的一系列操作

```
@Override
public void onSurfaceChanged(GL10 gl, int width, int height) {
    //摄像头开启预览
    mCameraHelper.startPreview(mSurfaceTexture);
    //可以去获取摄像头数据了
    mScreenFilter.onReady(width,height);
}
```



```
@Override
public void onDrawFrame(GL10 gl) {
    //配置屏幕
    //1.清理屏幕
    GLES20.glClearColor( red: 0, green: 0, blue: 0, alpha: 0);
    GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT);

    //把摄像头的数据先输出来
    //更新纹理
    mSurfaceTexture.updateTexImage();
    //获得变换矩阵
    mSurfaceTexture.getTransformMatrix(mtx);
    mScreenFilter.onDrawFrame(mTextures[0],mtx);

}

/**
 * 当有一个可用帧时调用
 * @param surfaceTexture
 */
@Override
public void onFrameAvailable(SurfaceTexture surfaceTexture) {
    //因为使用的是按需渲染，所以当有一个可用帧的时候，就去请求渲染一次
    mView.requestRender();
}
```

创建着色器

AS里面有个插件可以支持GLSL的高亮显示，在plugin里面搜索 GLSL Support

顶点着色器

```
// 把顶点坐标给这个变量，  确定要画画的形状
attribute vec4 vPosition;
//接收纹理坐标，接收采样器采样图片的坐标
attribute vec4 vCoord;
//变换矩阵，  需要将原本的vCoord (01,11,00,10)  与矩阵相乘 才能够得到 surfacetexure(特殊)的正确的采样坐标
uniform mat4 vMatrix;
//传给片元着色器 像素点
varying vec2 aCoord;
void main(){
    //内置变量 gl_Position ,我们把顶点数据赋值给这个变量 opengl就知道它要画什么形状了
    gl_Position = vPosition;
    // 进过测试 和设备有关
    //必须是 矩阵 * 纹理坐标
    aCoord = (vMatrix * vCoord).xy;
    //aCoord =  vec2((vCoord*vMatrix).x,(vCoord*vMatrix).y);
}
```


片元着色器

```
#extension GL_OES_EGL_image_external : require
//SurfaceTexture比较特殊
//float数据是什么精度的
precision mediump float;

//采样点的坐标
varying vec2 aCoord;

//采样器
uniform samplerExternalOES vTexture;

void main(){
    //变量 接收像素值
    // texture2D: 采样器 采集 aCoord的像素
    //赋值给 gl_FragColor 就可以了
    gl_FragColor = texture2D(vTexture,aCoord);
}
```


这里我们需要画的是矩形，也就是两个三角形，给了4个点的坐标，这4个点，会执行4次顶点着色器，依次将顶点传递给gl_Position，当4个点都传递完成之后，会进行光栅化，将一个矩形变换成一个个的片元，然后再去使用gl_FragColor去着色

因为SurfaceTexture是Android中的，并不是OpenGL的，所以需要使用额外扩展的采样器samplerExternalOES，而不是普通的sample2D采样器，在使用samplerExternalOES时候，需要再添加一句：#extension GL_OES_EGL_image_external : require