GPU主要分为三层

1.通过sample中activity和tools类调用选择滤镜，主要提供界面和交互

2.GPUImage、GPUImageView和Renderer类作为操作和刷新界面，处理获取的滤镜效果。

3. Filter类作为结果，返回给调用的tools类，通过第二层中的类进行处理。

Sample中调用系统相册获取相片和相机，两者都是需要用到GLSurfaceView。相机布局使用的是GPUImage，它包含GLSurfaceView和加载纹理、以及对相机的一些操作；使用相片通过继承FrameLayout实现布局，本质也是调用GPUImage对象，另外实现了对OpenGL的一些操作。每次需要更新时，调用setXXX方法设置参数，然后通知renderer类进行刷新

实现OpenGL和相机功能需要实现的接口Renderer和PreviewCallback，这些在GPUImageRenderer一并实现。

相片的纹理通过获取uri、file转换成bitmap，生成纹理；相机则是使用PreviewCallBack生成预览帧使用jni转换成intbuffer处理成纹理。

相机通过PreviewCallback接口setPreviewCallback可添加额外的效果到每一帧预览界面，相片则通过Renderer流程实现渲染。

GPUImageFilter第一层子类有五个分支，都是通过继承实现。

GPUImageTwoInputFilter:这个类作为混合效果的父类，本身并没有作为一个可实现的滤镜类，继承自这个类的子类实现的应该都是混合效果的滤镜。

混合的两张纹理，一张是当前相片或者相机预览生成的纹理，另外一张是随便一张图片。

第一张纹理在调用相片是以uri处理成bitmap，再转成纹理；在使用初始化混合的滤镜类的时候需要紧接创建第二张纹理。

在子类中重新编写混合的顶点着色器和片元着色器，混合两张纹理输出效果。

使用相机类似。

GPUImageFilterGroup:这个类主要起处理的作用。应该是对于多个滤镜叠加的而成的处理。通过使用FBO依次将每个效果绘制在FBO，下一次绘制的时候获取上一个FBO的纹理，叠加效果。

当使用相片时，会调用renderer里面的onSurfaceCreate、onSurfaceChange、onDrawFrame。在onSurfaceChange中调用onOutputSizeChange获取surfaceview宽高，创建FBO。在onDrawFrame调用了onDraw，子类滤镜就会调用FilterGroup的onDraw依次绘制滤镜效果，叠加处理。

使用相机的时候同理。

3x3TextureSamplingFilter:主要对一个像素点取样多了上中下三个位置。子类的片元着色器再通过这些额外的位置对像素处理。

ColorMatrixFilter以及直接继承GPUImageFilter的子类，没有细看。

1.GPU以继承的方式去实现每一个滤镜，灵活性拓展性比较好。在我的demo中，实现多种画笔效果以及每种画笔需要的program处理也是使用继承去增加新的种类跟继承滤镜类似，这部分代码开始主要参考书上的写法，每个部分比较分散。其画笔效果在绘制时需要绑定program，将可以像GPU归纳到一个基类去实现的工作分得太开，需要增加新类的时候反而难。对于这部分可以将画笔效果以及其对应的program写成一个类，将每个program跟随每个画笔。

另外以工厂模式去创建每一个当前需要的类，便于对每个类产生的变量等数据进行管理。在demo中由于对每个笔画的纹理都在renderer里一并创建，并没有很好的对所有的纹理program删除操作，可以将每个画笔对应的纹理等跟随每个画笔的生命周期管理。

2.GPU创建FBO是当surfaceview改变调用onSurfaceChange，获取到surfaceview新的宽高。

我的之前也是在onSurfaceChange重新获取宽高去设置FBO，后面重新写了布局，可以在activity创建的时候就计算出surfaceview会占据的宽高，便在onSurfaceViewCreate的时候就创建了FBO。主要是为了屏幕适配。