# [Android中直播视频技术探究之—摄像头Camera视频源数据采集解析](http://www.wjdiankong.cn/android%e4%b8%ad%e7%9b%b4%e6%92%ad%e8%a7%86%e9%a2%91%e6%8a%80%e6%9c%af%e6%8e%a2%e7%a9%b6%e4%b9%8b-%e6%91%84%e5%83%8f%e5%a4%b4camera%e8%a7%86%e9%a2%91%e6%ba%90%e6%95%b0%e6%8d%ae%e9%87%87%e9%9b%86/)

**一、前言**

在视频直播中一般都是两种视频数据源，一个是摄像头数据，一个是录制桌面数据，而一般来说美女妹子直播都是来自于摄像头数据，游戏直播都是录制桌面数据的，那么今天就来看看第一个数据源数据采集分析，Android中使用摄像头的场景很多，在没有直播这个行业出现之前，之前用到摄像头的最多就两个场景，一个是二维码扫描，一个是美颜拍照类的应用。那么这里就来看看Android中的摄像头的用法，以及如何进行数据采集进行数据的二次加工。

**二、知识点概况**

本篇文章主要通过如下几个方向去介绍Android中的摄像头Camera知识：

1、Camera摄像头的基本操作

2、Camera摄像头的前置和后置区分

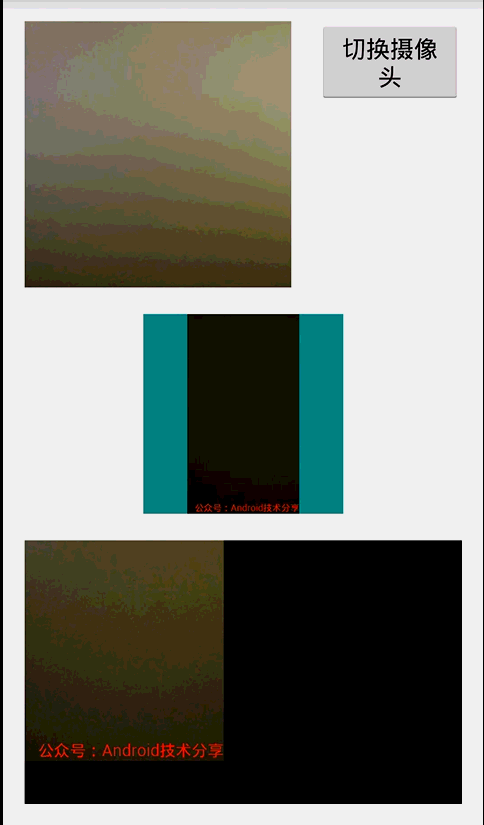
3、Camera摄像头的数据格式

4、Camera摄像头方向和数据尺寸

5、Camera摄像头的对焦拍照

6、Camera摄像头的数据采集以及二次加工

先来看看一张效果图：



**三、知识点详解**

通过上面的效果图，可以看到，可以切换前置和后置摄像头，可以对焦拍照，可以加水印效果。下面就来一一介绍内容

**第一、Camera摄像头的基本操作**

Android中使用一个摄像头其实很简单，首先需要在AndroidManifest.xml中声明权限：

然后代码中进行初始化操作即可：



初始化操作比较简单，就几步：

1、第一步：打开摄像头，使用open方法

这个方法有两种形式，一种是无参数形式的，默认打开是后置摄像头，还有一种方式是有参数像是，可以通过传递的参数来决定打开是前置还是后置摄像头，0代表后置摄像头，1代表前置摄像头。

2、第二步：设置摄像头的预览数据界面

可以不进行设置的，如果预览一般有两种方式，一种是调用setPreviewDisplay方法设置SurfaceHolder，也就是和SurfaceView进行绑定了，还有一种就是调用setPreviewTexture方法设置SurfaceTexture的，这个就和GLSurfaceView以及TextureView绑定了，这两种方式在介绍视频直播的基础知识的时候已经介绍了，还不了解的同学可以点击这里：[基础知识介绍](http://www.wjdiankong.cn/android%E4%B8%AD%E7%9B%B4%E6%92%AD%E8%A7%86%E9%A2%91%E6%8A%80%E6%9C%AF%E6%8E%A2%E7%A9%B6%E4%B9%8B-%E5%9F%BA%E7%A1%80%E7%9F%A5%E8%AF%86%E5%A4%A7%E7%BA%B2%E4%BB%8B%E7%BB%8D/)。后续如果要做美颜效果的话GLSurfaceView用的就比较多了，因为他本身集成了OpenGL的功能，而且二次处理的数据可以进行回显的。

3、第三步：获取到Camera.Parameters参数信息

通过getParameters方法获取摄像头已有的参数信息，然后进行相关设置，比如尺寸大小，方向，数据格式等信息。

4、第四步：在把添加好的参数信息设置回去，调用startPreview开始预览效果了

同样的摄像头的销毁方法也很简单：



也就四步：

1、第一步：将摄像头的预览清空

2、第二步：停止预览效果

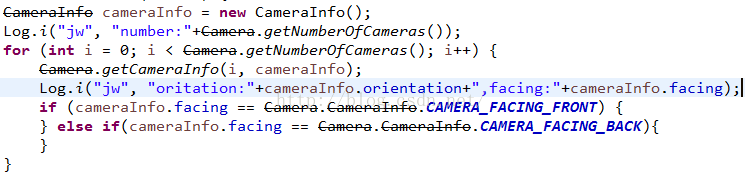
3、第三步：释放摄像头

因为系统默认只能同时开启一个摄像头不管是前置摄像头还是后置摄像头，所以不用的时候一定要释放

4、第四步：置空摄像头对象

**第二、Camera摄像头的前置和后置区分**

Android中的摄像头Camera是区分前置和后置的，所以这里就要做一个前置和后置摄像头的切换功能了，我们可以通过一个方法来获取当前系统的摄像头个数，以及摄像头的信息：

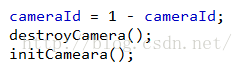


这个方法先获取当前设备中有多少个摄像头，一般现在设备都是两个，一个是前置，一个是后置的，我们得到信息之后在打印看看效果：

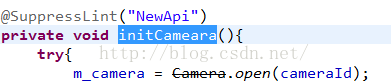
http://img.blog.csdn.net/20160728212214381?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

看到了，只有两个摄像头，而且通过默认的旋转角度得知，后置摄像头是90，前置是270，他们始终相差180度，不过这些和后面要说到的设置摄像头方向的效果没关系的。

那么切换摄像头Android中的做法是把之前的摄像头先销毁在重新初始化下一个摄像头：

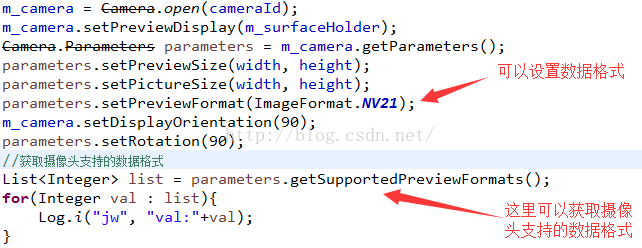


然后用一个全局的变量来记录当前是前置还是后置，在初始化方法中直接使用这个变量：



**第三、Camera摄像头的数据格式**

Android中摄像头的数据格式是指原生的每一帧数据，他是有指定格式的，而这些原生的每一帧数据可以有两个地方获取，一个是添加回调，在onPreviewFrame(byte[] data, Camera camera)回调方法中获取，还有一个就是拍照的时候回调方法：onPictureTaken(byte[] data, Camera camera)，当然如果想处理数据的话，肯定是在第一个回调方法中去进行操作了，第二种必须在拍照的时候把当前拍照的一帧数据拿到，这明显不靠谱。不管是那种获取一帧数据，这些都是有一定格式的，如果要后期处理的话，那么必须要做一次格式转化，Android中的摄像头数据的格式有两种，可以进行设置，当然直接使用代码可以获取到摄像头可以支持的数据格式：在Camera.Parameters类的getSupportedPreviewFormats方法即可获得：



看一下运行结果：



有两种格式，这里打印出来的是一个int值，如何查看这个int值对应的格式呢？可以去ImageFormat类中去查找：

第一种格式：

http://img.blog.csdn.net/20160728183836873?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

这个就是17对应16进制值，也就是ImageFormat.NV21格式的

第二种格式：

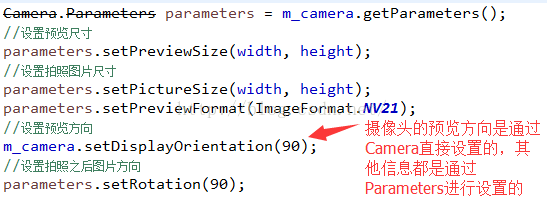
http://img.blog.csdn.net/20160728183904012?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

这个就是842094169对应的16进制值，也就是ImageFormat.YV12格式的。

上面了解了摄像头只支持这两种数据格式，那么我们后续肯定需要做数据处理，数据处理一般是和ARGB，以及后续的视频编码格式YUV420之间的转化，不过还好的是，这个转化格式网上已经有很多的，而且后续看到一些美颜功能都是处理这些数据格式的，同时这些操作最好放到native层做，因为效率会高一些。

**第四、Camera摄像头方向和数据尺寸**

Android中摄像头如果我们想要得到预期的数据的话，那么方向和尺寸非常关键，Camera中提供了一些方法可以进行这些参数的设置的，首先来看一下摄像头的方向问题：

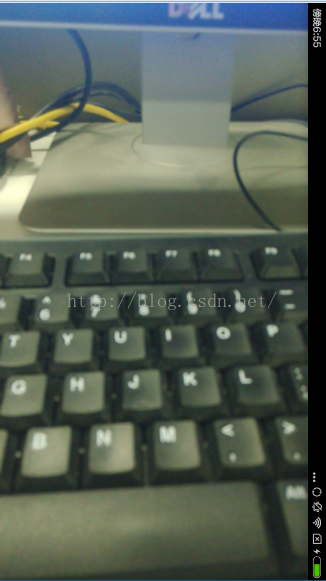


我们看到这里有两个方法可以来设置摄像头的方向信息，一个是Camera类本身的setDisplayOrientation方法，一个是Camera.Parameters类的setRotation方法，那么这两个方法有什么区别呢？

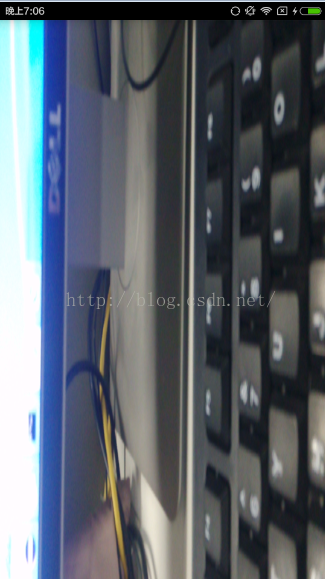
首先第一个方法：setDisplayOrientation方法是设置摄像头数据预览的方向的

就是我们用一个SurfaceView作为预览界面，在界面看到的方向。

Android中默认的预览方向是：横屏，旋转度是：0，所以如果你把当前Activity设置成横屏，不调用这个方法的话，效果是正常的：



但是，如果现在不设置横屏，因为Activity默认是竖屏的：



看到了效果，摄像头的预览方向没有发生变化的，所以这时候就需要调用setDisplayOrientation进行设置了，这个方法是按照逆时针旋转的，所以如果想让预览方向变正的话，就需要逆时针旋转90度即可：



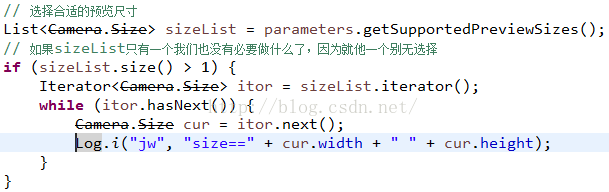
好了，到这里就知道了Camera的预览方向设置了，默认是：横屏方向，旋转度是0，如果想竖着拍摄的话，需要逆时针旋转90度即可。

还有一个设置方向，就是设置拍照的图片方向方法：setRotation这个方法的原理和上面的预览差不多，默认是：横屏方向，旋转度是0，如果想竖着拍照的话，需要逆时针旋转90度就可以了，但是这个方法的作用是设置Camera在拍照之后图片方向的，关于拍照后面会介绍。

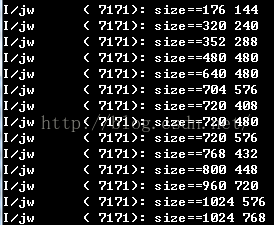
说到这里感觉还有一个方向不对，就是原生的每一帧数据的方向，的确，这里有一个问题就是我们在后面处理每一帧数据的时候方向是横向0旋转度的效果，而且没有方法可以进行设置，这个就比较蛋疼了，不过还好，因为我们后续再拿到每一帧数据肯定要做处理，到时候再做一次旋转就可以了。

说完方向问题，下面继续来看尺寸问题：

设置尺寸大小也有两个方法，这两个方法都是在Camera.Parameters类中的：setPreviewSize和setPictureSize，到这里就知道了这两个方法其实和上面方向的两个方法是一致的，第一个方法setPreviewSize是设置视频的预览尺寸的，第二个方法setPictureSize是设置拍照之后的图片尺寸大小的，不过这里有一个好处就是每一帧原生数据的尺寸可以获取到了，其实就是和预览的尺寸保持一致的，不像上面的方向不能设置。关于尺寸问题，其实是有指定限制的，同样可以通过一个方法来获取Camera所支持的尺寸大小：getSupportedPreviewSizes



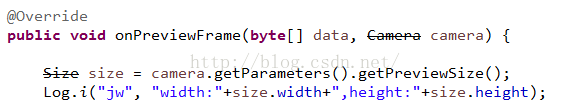
运行结果：



这是我的设备摄像头支持这些尺寸的，我们在设置尺寸大小的时候会在这里进行选择的，比如这里选择了720\*480尺寸，我们看一下效果：



看到摄像头采集的数据没有失真，但是有一个问题就是压缩了感觉，这个原因也很简单，因为720\*480的，之前说过这些尺寸是按照横屏定义的，所以我们现在是竖屏的，可以把尺寸调换位置就可以了，我们在每一帧数据中打印log：



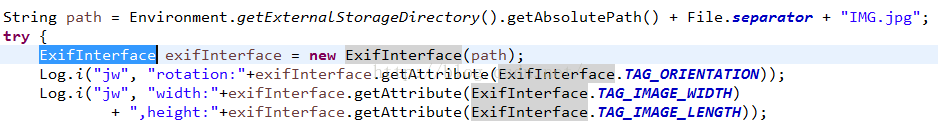
看一下结果：

http://img.blog.csdn.net/20160728201233305?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

这里的每一帧数据的尺寸就是上面设置的预览数据的。

下面来看一下设置拍照图片的尺寸，这里设置400\*400：setPictureSize(400, 400);

我们拍照，然后把图片保存到本地，在使用ExifInterface类来读取图片的元数据，这个类很重要的，可以读出图片的所有详细信息，比如在哪里拍的，什么时候拍的，方向，尺寸等信息，也就是传说中的图片的exif信息。



打印结果如下：

http://img.blog.csdn.net/20160728201616960?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

结果尺寸就是我们上面设置的拍照图片的尺寸大小。

好了到这里，我们知道了Camera也有两个方法可以设置尺寸大小，一个是设置预览的尺寸同时这个尺寸可以同步到每一帧数据的尺寸，还有就是拍照的尺寸大小。

**第五、Camera摄像头的对焦拍照**

再来看看Camera如何进行对焦拍照功能，这个其实就是一个简单的一个方法即可，在Camera类中的takePicture方法，这个方法的定义：takePicture(ShutterCallback shutter, PictureCallback raw, PictureCallback jpeg)

看到了，就是三个回调接口：

第一个回调接口用于：拍照之前的工作，比如现在系统的拍照有一个提示声音，我们当然也可以自己定义一个，就可以在这个回调接口中进行播放我们想要的提示声音即可。一般这个接口不怎么用。

第二个回调接口用于：原生的数据，就是当你拍照的那一帧数据的原生格式也就是NV21或者是YV12格式的。

第三个回到接口用于：直接一张图片数据，可以直接将数据保存成一张图片即可，无需进行数据格式转化。

代码很简单：

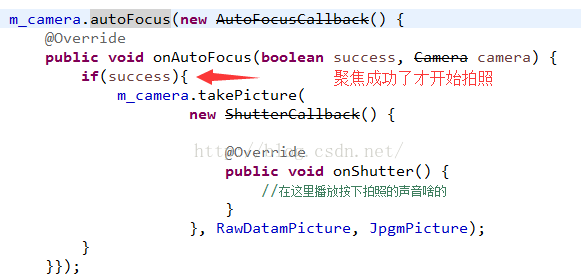


因为这里只关心图片，所以实现了最后一个接口：



拿到data数据，直接保存图片即可。然后在读取这张图片的exif信息。

但是我们在拍照的时候这么做还是有一个问题的，那就是拍出来的图片会模糊，因为有抖动效果，所以这里需要做一个聚焦效果，这个Camera也是有一个方法：autoFocus，同样也是一个回调接口：autoFocus(AutoFocusCallback cb)：



这里只有聚焦成功了之后才会进行拍照的。通过前面知识点可以知道，通过setPictureSize方法可以设置这个图片的大小尺寸的。通过setRotation方法可以设置图片的方向，而且这里还有一个点需要注意的是：拍完照之后需要再次调用Camera的startPreview方法，不然SurfaceView画面就停留在了当前页面了。

**第六、Camera摄像头的数据采集以及二次加工**

说完上面的所有知识之后，接下来这个知识点就是本文的重点，也是后续视频直播推流的重点核心，就是摄像头Camera的数据采集和二次加工。后续推流会在这里拿到每一帧数据进行传递，每一帧数据进行美化加水印都是在这里做。

Android中的摄像头Camera提供了两个方式回调接口来获取每一帧数据：

第一种方式：setPreviewCallback方法，设置回调接口：PreviewCallback

在回调方法：onPreviewFrame(byte[] data, Camera camera) 中处理每一帧数据

第二种方式：setPreviewCallbackWithBuffer方法，同样设置回调接口：PreviewCallback，不过还需要一个方法配合使用：addCallbackBuffer，这个方法接受一个byte数组。

第二种方式和第一种方式唯一的区别：

第一种方式是onPreviewFrame回调方法会在每一帧数据准备好了就调用，但是第二种方式是在需要在前一帧的onPreviewFrame方法中调用addCallbackBuffer方法，下一帧的onPreviewFrame才会调用，同时addCallbackBuffer方法的参数的byte数据就是每一帧的原数据。所以这么一看就好理解了，就是第一种方法的onPreviewFrame调用是不可控制的，就是每一帧数据准备好了就回调，但是第二种方法是可控的，我们通过addCallbackBuffer的调用来控制onPreviewFrame的回调机制。

注意：

因为第二种方式在调用的时候有点注意的地方：

1》在调用Camera.startPreview()接口前，我们需要setPreviewCallbackWithBuffer，而setPreviewCallbackWithBuffer之前我们需要重新addCallbackBuffer，因为setPreviewCallbackWithBuffer 使用时需要指定一个字节数组作为缓冲区，用于预览图像数据 即addCallbackBuffer，然后你在onPerviewFrame中的data才会有值。  
2》从上面看来，我们设置addCallbackBuffer的地方有两个，一个是在startPreview之前，一个是在onPreviewFrame中，这两个都需要调用，如果在onPreviewFrame中不调用，那么，就无法继续回调到onPreviewFrame中了。

本文直说第一种方式，下面就来看看如何通过获取摄像头的每一帧数据，进行二次处理：



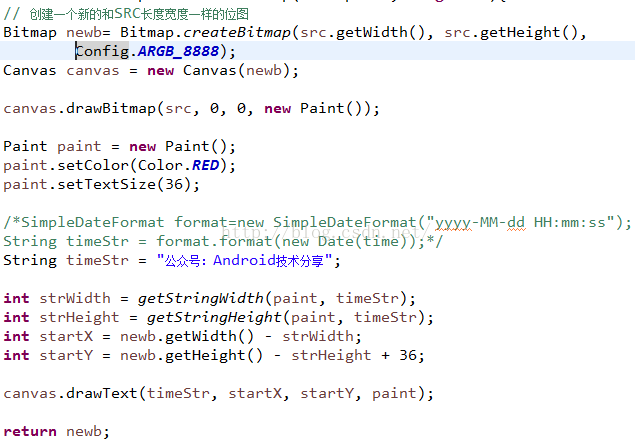
代码处理其实也很简单，主要分为三步：

1、第一步：数据转化

把NV21/YV12格式转化成ARGB格式在生成一张图片，这里有两种方式，一种是直接使用系统的YuvImage类进行转化即可，要传递图片的尺寸，这个可以通过getPreviewSize方法获取到。还有一种方式就是使用网上的转化工具类即可，不过像这种工具类转化工作应该放到native层去做，效率高很多。

2、图片二次处理

上面说到了，Camera中虽然有两个可以设置方向的方法，但是一个是设置预览方向的方法，一个是设置拍照图片的方向，没有设置每一帧原始数据的方向，所以这里我们转化得到的一张图片应该是横着的，因为Camera默认的模式是：横屏+0度旋转，所以这里如果想让图片方向正确的话，应该做一次图片旋转，而且前置摄像头和后置摄像头旋转的角度不一样，不过好理解的是，前置摄像头和后置摄像头就相差180。旋转之后得到正确方向的图片之后，开始添加水印效果了，添加水印效果非常简单：



直接使用Canvas画布即可。

3、处理之后的图片数据加工

上面处理了数据之后得到我们想要的图片数据，那么下面就可以进行后续操作了，这里是直接进行展示，后续一篇文章会介绍把这数据推流到服务端。本文先不介绍了。

这里看到了，核心的方法就是在这里，我们先处理数据格式，然后加上我们想要的效果，比如水印，美颜，因为美颜不是本文的重点，为什么呢？因为美颜需要用到OpenGL相关的知识，后续会详细介绍如何使用OpenGL来处理图片。有了这个知识点，很多工作都可以做了：

1、美颜相机

首先得到每一帧图片之后，进行处理美白，这里还可以做人脸识别来添加美颜贴纸，这个就需要人脸识别算法了。不过这里需要注意的一点是，美颜相机预览的时候肯定使用的GLSurfaceView的，因为我们在之前的一篇基础知识：[点击这里查看](http://www.wjdiankong.cn/android%E4%B8%AD%E7%9B%B4%E6%92%AD%E8%A7%86%E9%A2%91%E6%8A%80%E6%9C%AF%E6%8E%A2%E7%A9%B6%E4%B9%8B-%E5%9F%BA%E7%A1%80%E7%9F%A5%E8%AF%86%E5%A4%A7%E7%BA%B2%E4%BB%8B%E7%BB%8D/) 了解到了GLSurfaceView和SurfaceView以及SurfaceTexture的区别就是，SurfaceView无法将二次处理之后的数据再次回显到界面上了，而我们看到美颜相机在使用的时候，选着一种滤镜的时候，画面会立即改变的，其实这个就是使用GLSurfaceView来做的，因为GLSurfaceView首先可以结合OpenGL，其次他可以把二次处理之后的图片在回显到界面中的。

2、二维码扫描

得到每一帧数据之后立马识别二维码，识别成功立即退出即可。

**项目案例下载：**[**http://download.csdn.net/detail/jiangwei0910410003/9589158**](http://download.csdn.net/detail/jiangwei0910410003/9589158)

**四、技术总结**

1、摄像头Camera的基本操作：初始化操作和销毁操作

1》初始化操作：调用open方法直接打开摄像头，然后设置预览载体，在设置摄像头的一些参数信息，最后设置每一帧的回调接口，开始预览效果。

2》销毁操作：置空每一帧的回调接口，停止预览效果，释放摄像头

2、摄像头Camera的前置和后置效果

Android中的前置摄像头和后置摄像头切换是通过把之前的一个摄像头释放，然后在重新初始化下一个摄像头，同时用一个全局的变量来标志当前摄像头的状态即可，使用open带有参数的形式来决定打开那种摄像头。

3、摄像头Camera的数据格式处理

Android中的Camera的数据格式是可以设置的，但是摄像头只支持两种格式NV21和YV12，所以我们在后续的数据二次处理就需要做数据格式转化，一般都是把数据转化成ARGB格式或者是视频编码的YUV420格式。

4、摄像头Camera的尺寸和方向设置

1》Android中的Camera可以获取到当前所支持的尺寸大小，但是需要注意的是，因为Camera默认的方向模式是：横屏+0度旋转，所以宽度\*高度尺寸是针对于横屏来说的，所以看到这些尺寸都会发现宽度比高度值大。如果设备是竖屏的话，我们需要做一次尺寸调换。同时可以支持两种方式设置尺寸大小的，一个是可以设置预览的尺寸大小，而这种大小将会同步到到每一帧原始数据的尺寸大小的，还有一个就是可以设置拍照之后的图片大小。

2》Android中的Camera的可以支持两种方向设置的，一种是预览方向设置，还有一种是拍照之后的图片方向设置，这里我们学习到了图片的exif信息处理。但是唯独没有每一帧原始数据的方向设置，所以我们后续再处理每一帧数据的时候需要手动的做一次方向旋转，旋转的时候还要区分前置摄像头和后置摄像头的旋转角度。

5、摄像头Camera对焦拍照

Android中Camera可以通过autoFocus方法设置对焦回调方法，然后在使用takePicture方法设置回调方法获取拍照之后的图片数据，可以直接保存成图片即可。无需数据格式转化。

6、摄像头Camera的原生每一帧数据采集

这个知识点是最重要的，是后续推流和编码的核心点，他能够获取到摄像头的每一帧数据，我们可以在这里做二次处理，比如把原生的NV21/YV12数据格式转化成ARGB格式，然后添加水印效果。

**五、用途**

了解到了上面的知识点之后，我们可以做的事情就很多了，比如美颜相机我们是可以实现的，二维码扫描也是可以的。如果想做美颜的贴纸功能就需要人脸识别算法，如果想使用更多的滤镜效果，那么就需要强大的OpenGL来做图片处理。

**六、总结**

本文主要介绍了Android中视频直播中的一种视频数据源：摄像头Camera的使用和后续推流的准备工作，下一篇文章我们将开始重点介绍如何将摄像头采集的数据进行二次处理，然后推流到服务端，在拉流进行观看。