WebRTC源码分析四：视频模块结构

本文在上篇的基础上介绍WebRTC视频部分的模块结构，以进一步了解其实现框架，只有了解了整体框架结构，对局部算法修改才能够胸有成竹。

一、对外接口

对外接口有ViEBase，ViECapture，ViECodec，ViEEncryption，ViEFile，ViEImageProcess，ViENetwork，ViERender和ViERTP_RTCP。主要接口有：

1）ViEBase：负责创建和管理视频会话链路；

2）ViECapture：设置采集设备和参数；

3）ViEImageProcess：视频画面处理配置；

4）ViENetwork：通信端口设置；

5）ViERender：渲染设备选择与设置。

与音频类似，对外接口都是通过VideoEngine对象来获取：

```
ViEInterfaceXX*pInterface = ViEInterfaceXX:GetInterface(pVideoEngine);
```

二、模块组成

模块之间的关系如图1所示，红色标明的模块与视频产生相关，VideoCaptureModule负责产生视频数据，通过VideoCaptureDataCallback和VideoCaptureFeedback回调传递给采集模块。粉红色标明的模块与视频采集相关，ViECapturer负责采集视频数据，封装成视频帧。通过ViEFrameProviderBase回调注册的ViEFrameCallback的接口。由于ViEEncoder实现了ViEFrameCallback接口，所以视频帧传递给ViEEncoder进行编码，VideoCodingModule一侧模块完成是视频帧的编码工作。此外ViEEncoder实现了VCMPacketizationCallback接口，当编码完视频后，调用该接口通过RtpRtcp模块发送视频帧。绿色标明的模块与视频的渲染相关。ViEChannel负责解码接收的视频帧，解码后通过ViEFrameProviderBase接口将视频帧传递给ViEFrameCallback，由于ViERenderer实现了该接口，所以最终传递到ViERenderer中，ViERenderer负责后续的渲染工作。

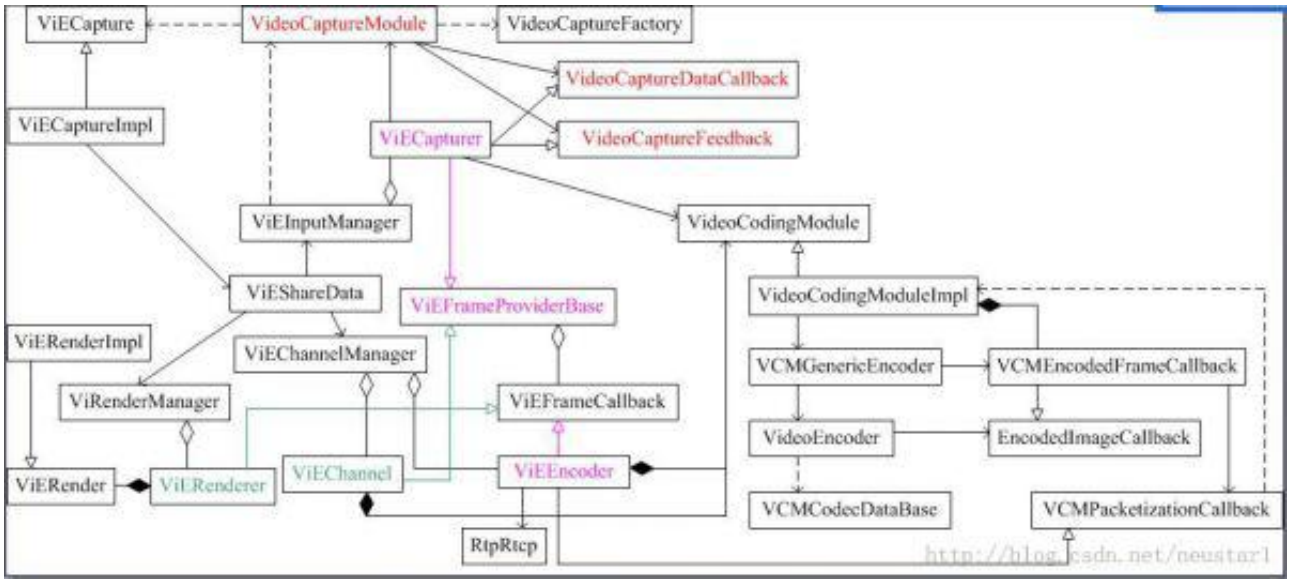

图1 模块关系

三、设计模式

WebRTC源码中存在许多类似的场景：

```
Class CallbackA

{

Virtual void Completed()=0;

}

Class B

{

Public:

        B(CallbackA*pCallbackA):Callback_(pCallbackA){}

        Void Exec()

{

    /执行某些操作*/

    Callback_->Completed();

}

}

Class A:public CallbackA
```

```
{
Public:
    Void DoThing()
{
        m_pB = new B(this);
        m_pB->Exec();
    delete m_pB;
}
Private:
    B* m_pB;
}
```

之所以使用这种方式，有两方面原因，一方面是层次划分的原因，CallbackA属于A类这一层的调用操作，但是它又必须在B类的某个方法后执行，所以使用回调。另外一方面为了可扩展，允许在现有的结构上实现更多功能。

由于没带摄像头，需要调试一会，后续给出视频通信的代码示例。。。如果有不对的地方欢迎讨论，多多学习！