DirectShow入门之动态构建FilterGraph PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/273/2021_2022_DirectShow_c97_273529.htm 摘要:本文档主要讲述了Filter graph的动态构建技术 动态的重新连接 在进行pin连接的时候，应用程序一般都要讲graph停掉。但是，一些filter支持pin的动态连接。 图1 如上图，我们想将Filter 2动态移走。有两个必要条件: (1)Filter 3 (pin D)必须支持IPinConnection接口(这个接口能够保证Filter在非Stopped状态下也能进行Pin的重连). (2)Filter1上的输出pin，也就是pin A，在重新连接的过程中必须要将发送数据的线程阻塞，在重连的时候，pin A和pin D中不允许数据的传输，如果“ 重连 ”是由Filter 1发起的(在Filter内部完成)，那么Filter 1要有内部机制来阻塞数据发送线程.如果“ 重连 ”由应用程序来完成，则要求Filter 1 (pin A)实现IPinFlowControl接口。 动态重连的一般步骤如下: (1)在Filter 1(pin A)上阻塞数据流线程。 (2)重连Pin A和Pin D，必要时插入新的Filter。 (3)再次启动Filter 1(pin A)上的数据发送线程。 下面先来看看第一步. (1)在Filter 1(pin A)上阻塞数据流线程。 IPinFlowControl::Block可以工作在同步和异步两种模式下。 如果在异步方式下使用Block函数，要首先创建一个win32事件，然后将这个事件句柄作为参数传递给block函数，这个函数会立即返回，然后调用WaitForSingleObject方法等待事件被触发。如下:// Create an eventHANDLE hEvent = CreateEvent(NULL, FALSE, FALSE, NULL).if (hEvent != NULL){ // Block the data flow. hr = pFlowControl->Block(AM_PIN_FLOW_CONTROL_BLOCK, hEvent). if (SUCCEEDED(hr)) { // Wait for the pin to finish.

DWORD dwRes = WaitForSingleObject(hEvent, dwMilliseconds). }} 如何用同步的方式调用Block函数，只需给它传递一个NULL的参数即可。这样这个函数就会一直阻塞下去，直到pin完全做好准备可以传递sample，阻塞停止。如果filter处于paused状态，阻塞时间也许会无限延长，因此，不要在你的主线程中采用同步的方式调用这个函数，可以新开一个线程或者采用异步的方式。 (2)重连Pin A和Pin D，必要时插入新的Filter。 这里Pin的重连可以使用IGraphConfig::Reconnect或IGraphConfig::Reconfigure(IGraphConfig接口可以从Filter Graph Manager上获得)。Reconnect比Reconfigure使用起来要简单，它主要作如下几件事: 1、将Filter 2置于Stopped状态，然后将其移走. 2、如果需要，可以在filter graph中增加新的Filter. 3、重新连接相关的各个Pin. 4、将新加入的Filter置于Paused或Running状态，以使其与Filter Graph同步。示例如下:pGraph->AddFilter(pNewFilter, L"New Filter for the Graph").pConfig->Reconnect(pPinA, // Reconnect this output pin... pPinD, // ... to this input pin. pMediaType, // Use this media type. pNewFilter, // Connect them through this filter. NULL, 0). 实际应用中，如果你觉得Reconnect不够灵活，还可以改用Reconfigure方法。这个方法可以通过应用程序中的回调函数来重新连接pin。使用Reconfigure方法，你必须在你的应用程序里实现IGraphConfigCallback接口. 在Reconfigure调用之前，还要如前所述，阻塞数据发送线程，然后通过以下步骤将海没有处理得数据发送下去: 1、调用filter最末端的输入pin上的IPinConnection::NotifyEndOfStream。在这个例子里就是pin D。给这个pin传递一个win32事件 2、然后在和上游需要阻塞

发送数据的filter的输出pin相连的输入pin上调用IPin::EndOfStream方法，这个例子，上游最远端得数据输出pin是pin A，那么和它相连的输入pin就是pin B了，就调用pin B上的IPin::EndOfStream方法。 3、等待触发事件。当Pin D接收到一个end of stream通知的时候，表明graph中已经没有数据流在传递了，可以安全的进行pin的重新连接了。 这些处理IGraphConfig::Reconnect会自动给我们完成 。 (3)再次启动Filter 1(pin A)上的数据发送线程。 只需调用IPinFlowControl::Block，如下:pFlowControl->Block(0, NULL). 过滤器链(Filter Chains) 首先要弄明白什么是Filter Chain。见下图: 图2 1、Filter Chain是相互连接着的一条Filter链路，并且链路中的每个Filter至多有一个Input pin，至多有一个Output pin. 2、这条Filter链路中的数据流不依赖于链路外的其他Filter。 如上图中，AB，CD，FGH，FG，GH都是Filter Chain，同时Filter链也可以只包括一个filter，因此A,B,C,D,E,F,G也都是独立的链，因为E含有两个输入pin，因此任何含有E的都不是Filter Chain。 Filter Chain通过IFilterChain接口来操作的，该接口可以从Filter Graph Manager上获得。 IFilterChain提供了下面的方法用来操作filter链: IFilterChain::StartChain 开始一个链条 IFilterChain::StopChain 停止一个链条 IFilterChain::PauseChain 暂停一个链条 IFilterChain::RemoveChain 将一个链条从graph中删除 并没有一个特殊的方法用来添加一个chain，它和正常的添加filter的方法一样，首先用IFilterGraph::AddFilter 在graph中添加一个filter，然后就是IGraphBuilder::Connect, IGraphBuilder::Render等诸如此类的方法。 当Graph在运行的时候，Filter Chain可以

在Running和Stopped状态之间切换.当Graph在暂停状态下，Filter Chain可以在Paused和Stopper状态之间切换。以上是Filter Chain仅有的两种状态转换。