

# Using the CMultiIOBaseFilter class

## Overview

The CMultiIOBaseFilter class follows the same design as the CTransformFilter class. The only difference is that it allows the implementer of the subclass to easily create a filter with multiple input and output pins and takes care of all the base work needed. This should make it easy for anyone used to writing standard transform filters to reuse this class as a basis for writing transform filters with x input pins and y output pins.

These base classes take care of:

- Memory management of input and output pins (Covering initial and subsequent requirements)
- Acceptable media type management on a per pin basis
- Providing an easy way to write transform filters with more than one input/output pin

Similarly to the CTransformFilter class, CBaseInputPin and CBaseOutputPin have been subclassed and further responsibility has been delegated to the CMultiIOBaseFilter class.

See the example StreamConcatFilter project for more information. This filter simply takes two stream inputs and concatenates them into one image. No error handling has been done and for testing I simply connected my webcam input to an infinite tee filter. I then connected the two (similar) outputs of the infinite tee filter to the StreamConcatFilter. The output was then connected to a video renderer which rendered the desired picture (See the graph in figure 1).

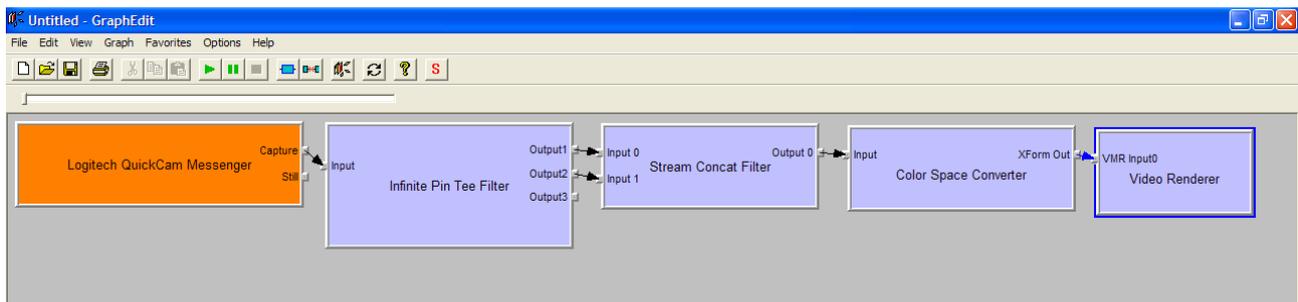


Figure 1:

Visual Studio 2005 solutions and projects have been provided. Should you be using an older version of Visual Studio, you'll have to create the projects yourself. Once the MultiIOBaseFilter static lib has been build, link this into the StreamConcatFilter project.

## Usage:

On the development the following steps are necessary to use this base class

### Create subclass of CMultiIOBaseFilter

Extend the base class as is illustrated in the example project.

### Call Initialise in your subclass constructor

This method calls the virtual method needed to initialise the acceptable types, subtypes and formats on the input and output pins of the filter, as well as initial number of input and output pins

### Override InitialNumberOfInputPins and InitialNumberOfOutputPins

The default number of input and output pins is 1. Override these methods if this is not suitable for your

application

## **Override InitialiseInputTypes and InitialiseOutputTypes**

Call the AddInputType and AddOutputType method with acceptable media types for your filter's input and output pins.

## **Override OnFullCreateMoreInputs and OnFullCreateMoreOutputs**

These methods determine whether new inputs/outputs are created once all available ones have been used.

## **Override DecideBufferSize**

This method is very similar to the CTransformFilter method which needs to be overridden. The extra parameter denotes which output pin the buffer size is being decided for.

## **Override GetMediaType**

This method again is very similar to the CTransformFilter equivalent. The extra parameter again denotes which output pin is being queried for its media type.

## **Linking:**

Be sure to link the DirectShow base class library into the projects. Once you've built the static MultiIOBaseFilter.lib lib, be sure to link this into your own custom filter DLL.

## **Contact:**

Please send comments, feedback, suggestions to [rtvc \(at\) meraka \(dot\) org \(dot\) za](mailto:rtvc@meraka.org.za)