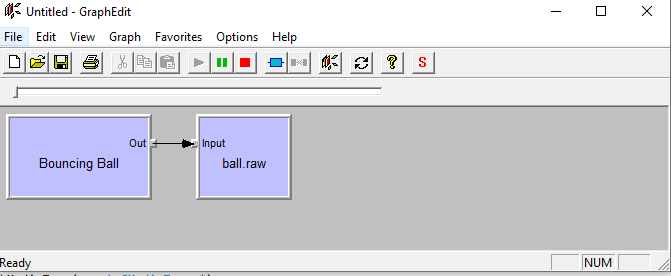
**Raw video player with direct show**

In this video I want to show you and share raw video player.

First thing, I don’t have raw video that I can show you, so let’s create raw video using two of the filter’s examples in DirectShow:

1. Ball
2. Dump



If you don’t have them In your computer yet installer already, download the DirectShow sdk , multimedia in Windows Sdk , compiled both examples and register it.

This video will not show you how, if you need help write in the comment.

While writing down, I let the graph running and create a big file for our raw video player



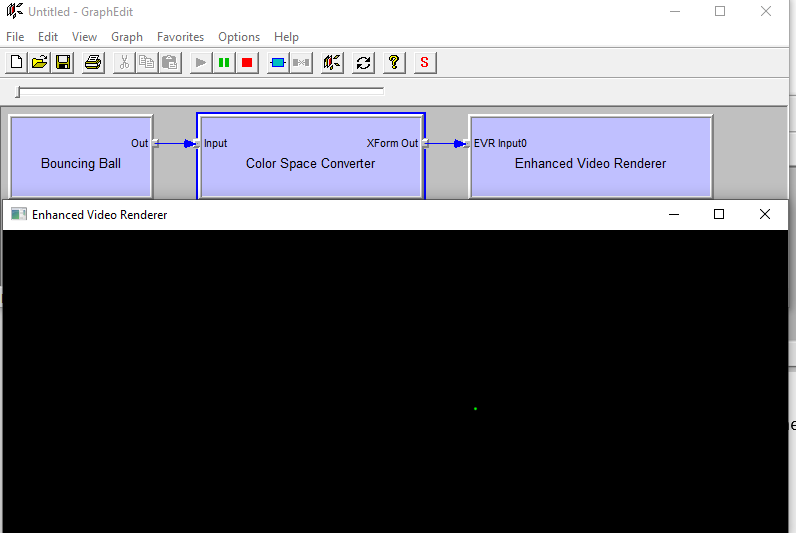
Video raw player is a Ball raw source but with modification

1. It has interface to load from file
2. It connects to video renderer and not to dump.

The same way we view the ball playing, we can make the filter load the raw file and inject it to the fillbuffer output pin.

We have the bouncing ball source code, so we can do what every we want.

Before we continue, let’s see what happened if we want to see the ball in action:

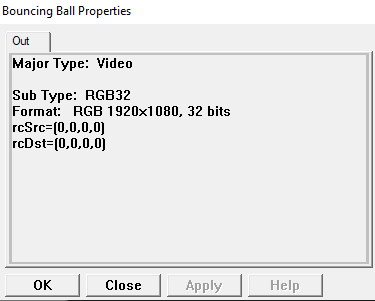


The original ball is yellow, here we see it green.

The output pin of the ball is now 24 bit , there for graph edit adds the color space convertor to move to 32 bit.

Which it did not do a good job, it change the colors.

We have the code, so we can compile to force the output pin to be 32 bit



HRESULT CBallStream::GetMediaType(int iPosition, CMediaType \*pmt)

{

CheckPointer(pmt,E\_POINTER);

CAutoLock cAutoLock(m\_pFilter->pStateLock());

if(iPosition < 0)

{

return E\_INVALIDARG;

}

// Have we run off the end of types?

if(iPosition > 1)

{

return VFW\_S\_NO\_MORE\_ITEMS;

}

VIDEOINFO \*pvi = (VIDEOINFO \*) pmt->AllocFormatBuffer(sizeof(VIDEOINFO));

if(NULL == pvi)

return(E\_OUTOFMEMORY);

ZeroMemory(pvi, sizeof(VIDEOINFO));

if (m\_pixelDepth == BI\_RGB && m\_bitCount == 32) // we force it here

{

// Return our highest quality 32bit format

// since we use RGB888 (the default for 32 bit), there is

// no reason to use BI\_BITFIELDS to specify the RGB

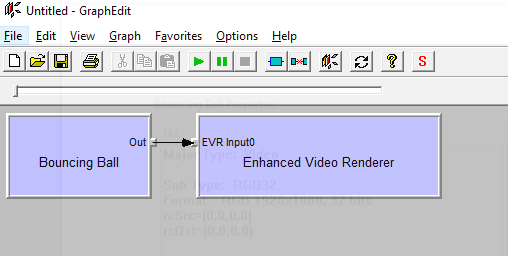
// masks. Also, not everything supports BI\_BITFIELDS

SetPaletteEntries(Yellow);

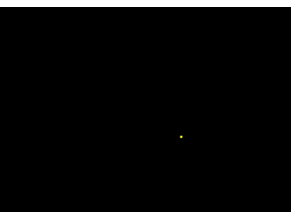
pvi->bmiHeader.biCompression = BI\_RGB;

pvi->bmiHeader.biBitCount = 32;

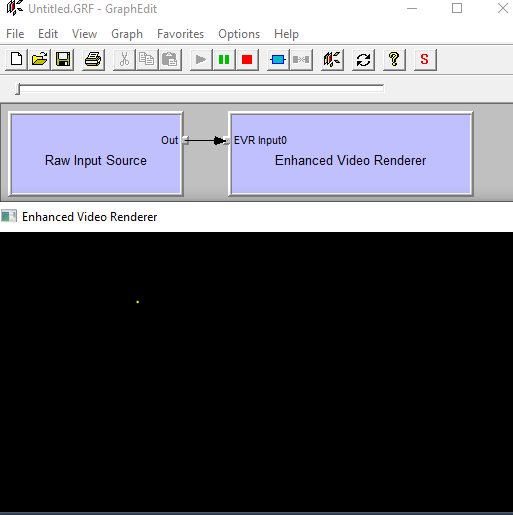
}



The result is a yellow ball running on the screen, which is the correct color



Here is our Raw input source in action:



The input is 24 bit and the output is 32 bit inside the filter:

We add additional loop from 24 bit to 32 bit

for (int i = 0; i < m\_pFilter->HD\_24\_bit; i += 3)

{

pData[j] = 0;

pData[j + 1] = pData24Buffer[i + 1];

pData[j + 2] = pData24Buffer[i + 1];

pData[j + 3] = pData24Buffer[i + 0];

j += 4;

}

The raw input source has interface:

static const GUID IID\_IRawInputSource =

{ 0x653520da, 0x11b7, 0x46c8, { 0x96, 0x4a, 0xa7, 0xd4, 0xd4, 0x43, 0x26, 0xe1 } };

typedef void(\_\_stdcall \* RawSourceDelegate)(int frameNumber, int frameCount);

DECLARE\_INTERFACE\_(IBoutechRawInputSource, IUnknown)

{

STDMETHOD(SetResolution)(int x, int y, int inFormatRes, int outFormatRes, INPUT\_RAW\_SOURCE\_FORMAT inputFormat) PURE;

STDMETHOD(SetFileName)(WCHAR \*flileName) PURE;

STDMETHOD(SetStartFrame)(long startFrame) PURE;

STDMETHOD(JumpToFrame)(long frameNumber) PURE;

STDMETHOD(Loop)(bool l) PURE;

STDMETHOD(PauseVideo)()PURE;

STDMETHOD(ResumeVideo)()PURE;

STDMETHOD(StepForward)()PURE;

STDMETHOD(StepBackward)()PURE;

STDMETHOD(RegisterRawSourceCallback)(RawSourceDelegate callback) PURE;

STDMETHOD(SetRawSourceFrameDelay)(int frameDelay) PURE;

STDMETHOD(ClearSequenceList)() PURE;

STDMETHOD(AddSequenceFile)(const WCHAR \*fileName) PURE;

};

The interface helps to configure the filter from host application in cpp

The sources of the filter can be found here:

<https://github.com/EliArad/RawVideoSourceFilter>

along with more than 75 repositories.