# Before you start

For manipulate media there are a number of open source frameworks, each with his own target. The three that I would recommend (With Delphi wrappers) are:

* FFMPEG – best for understanding and unwrapping, wrapping media streams and files from one hand and decoding encoding the streams. (Also used in all kinds of GStreamer plugins).
* OpenCV – best for unveiling what reality is actually in the media (Also used in some GStreamer plugins, filters).
* GStreamer – what is explained here, best for stream routing & filtering of media.

In my opinion, GStreamer is more complicated to master and if you can work with the above two you may be better off.

# Abstract

G2D is a bridge between GStreamer framework and Pascal. G2D enables Delphi (pascal) developers to use the GStreamer framework in their pascal program. By doing so G2D enables endless manipulation and uses of multimedia on Windows, Linux, Mac, ios and Android systems. Because GStreamer was unavailable for Delphi developers many professional multimedia project development used C, C++, Python and Java (that do have bridges to GStreamer) although in all other aspects Delphi would be their preferred choice.

# What is GStreamer

GStreamer is an open source (LGPL license) framework written in C (not C++) for manipulating Streaming Data.

In many aspects, it might seem to be using C++. However, its underlining framework is GObject that enables pseudo object oriented in C.

Streaming Data, is data coming from an endless source. Think of a video camera that is never stopped. The data will be streaming from that camera. Multimedia, video and audio are the most common streaming data, but there can be other kinds, like seismic measurements taken 100 times a second etc.

### *Basic concept of GStreamer*

You can think of GStreamer as a Lego game. You build whatever structure you want using all kinds of Lego Blocks. Like a Lego game, you can connect the different blocks one to the other. However, the GStreamer block is a pipe. As in Lego, you connect one piece of pipe to the other thus building the structure (pipeline) you want. Each pipe piece is called element or plugin. As an open source, there are endless types of elements. For example: plugin that show the video on the screen, plugin that receive video or audio from IP communication or from a file, plugin for filtering the streaming data, plugin that split and copy the stream into two streams etc.

You will usually chain (link) the elements one to the other. For example, you can link a video generator element to a filter element to an element that displays the video. So on screen you will see the video generated after it was filtered. To run a chain from a command line you can use a command like:

gst-launch-1.0.exe filesrc location=thesong.mp3 ! decodebin ! audioconvert ! pulsesin

Where gst-launch-1.0 is the program (launch program), and all the other stuff in the above line are arguments to the launch program. These arguments are basically, the names of the plugins. The launch program will use these plugins as a chain (pipeline) of elements. The “!” is a sign for the launch program to connect & link the plugins as a chain of pipes for the stream of data.

The plugins are built into DLL files. Each DLL file can contain one or more plugins. If you want to master the stream of data, you need two abilities:

1. To launch a pipeline (with a stream) from within your application.
2. To build your own plugins

Basically, you can work with one of the two above abilities. If you can launch a pipeline from your pascal program, you can use the endless plugins that are available to manipulate the stream. On the other side if you build your own plugin in Pascal, you can launch it from the command line. But you can use both abilities and run a stream through your plugin all within Pascal. The main part of G2D is the ability to easily write plugins & launch pipelines in Pascal.

To understand the GStreamer framework, and how to use it, you should refer to sources on the internet. Starting point for that can be:

<https://gstreamer.freedesktop.org/documentation/>

### *Understanding a Plugin*

As mentioned above a plugin is in a DLL that serves as a pipe (A Lego Block) for the GStreamer framework.

##### PlugIn

*Note: klass is a C pseudo class that is almost the equivalent of a C++, Java or Pascal class. The G2D wraps the klass into a Delphi class.*

Plugin is a Gst-element (klass) that is a piece of pipe. As mentioned above it can be one of three basic types

1. At the start of a pipeline were it gets the stream from a hardware device that can be a file on disk, a network from IP, a camera device, etc. then it is called a source plugin.
2. At the end of the pipeline were it sends the stream to a hardware device that can be a file on disk, a network from IP, the screen, etc. then it is called a sink plugin.
3. In the middle of the pipe than it is just called plugin.

##### Pipeline

Pipeline is a Gst-element (klass) that holds multiple plugins. The pipeline masters the flow of the stream. (If there is only one plugin in the pipeline, you do not need the pipeline and you can use the plugin as the pipeline)

##### Pads

Pads are also a Gst-element (klass) that are part of a plugin.

As explained above, a plugin is a piece of pipe. It has two ends (called Pads). There is one pad for receiving the data stream, called “sink pad” (it is a sink for the previous plugin). At the other end, a pad for sending the data stream to the next plugin called “src pad” (sink like a kitchen sink where you send the water stream to).

A plugin that is capable of handling Video stream probably cannot handle a Seismic measurements stream. You might build a video plugin that can only handle raw video and not compressed video or a video that can handle only video with width of 640 pixels X height of 480 pixels. Alternatively, some range of width and height. The receiving pad, the sink pad and the src pad should negotiate their capabilities with one another. This is called capability exchange and will be discussed later.

Note: do not confuse “src pad” with “src plugin”. A “src plugin” is the plugin that starts the chain so “src plugin” receives the data stream from some kind of hardware that can be a camera, or an IP communication or a file etc. In the same way, a “sink pad” is not a “sink plugin”. A “sink plugin” is the end of a chain. “Sink plugin” sends the data stream to some kind of hardware that can be the speakers of the computer for an audio stream, or a file etc.

##### Flow (We address Video streams – but true for all streams)

The flow of the data stream is divided into buffers (chunks of memory filled with the data from the stream). For example, in a raw video plugin, a buffer is usually one frame of the video. The frame is usually arranged in a 2-dimension array (like the screen itself – or a graphic window on the screen) where each cell holds the information of one pixel of the frame. Each cell of the matrix might be 3 bytes corresponding to Red Green & Blue (RGB) of that pixel. When video is flowing in 30 frames per second the src pad should receive 30 buffers per second.

The src pad is responsible for receiving the buffers in a continuous periodic synchronized manor and the sink pad is responsible for sending the buffers in the same continuous periodic synchronized manor. So src pad and sink pads must be synchronized in their buffer exchange. That is what they are built for.

The launch program (or procedure), calls the methods that operate in the plugin, usually the main thread of the program. The capability exchange methods are called only before streaming starts. In contrast, the operating of data stream buffers by src/sink and others are called in a continuous periodic synchronized manor.

# Be Aware

This G2D framework is in early stages of construction!!!

In this stage, it supports very limited operation.

Here is a partly list of limitations:

* Support only windows (tested on windows 10 desktop only)
* Support only programs compiled in 64 bit
* Support only desktops that installed the full GStreamer for windows “68\_64” (the 64-bit version). In addition, did not customize any directory places etc.
* Most of GStreamer function are not yet bridged to pascal. We are only at the beginning of the process.

# Installation

Before you can start programing with GStreamer in Delphi, you must install two main things:

1. Install the GStreamer framework.
2. Install the G2D that is the bridge between Delphi and GStreamer.

*Note: G2D should work with any pascal compiler but was not tested for that, only Delphi 10.3.3 was tested.*

### Installing GStreamer framework

Download from <https://gstreamer.freedesktop.org/data/pkg/windows/1.16.2/>   
choose two \*.msi files: “gstreamer-1.0-devel-msvc-x86\_64-1.16.2.msi” & “gstreamer-1.0-msvc-x86\_64-1.16.2.msi”  
After downloading be sure to install them (by Double click on both). When installing do not change anything (just press next… and finish) so they will be installed in the default directories.

### Installing GStreamer G2D

Open: <https://github.com/sharonido/Delphi_GStreamer> you should download the whole repository by the green download button, or if you have git installed in your system then from cmd line enter the clone command:

*git clone* [*https://github.com/sharonido/Delphi\_GStreamer.git*](https://github.com/sharonido/Delphi_GStreamer.git)

It is important to maintain the G2D internal structure. It does not matter where you decide to put this directory structure in your system.

##### Explanation

In bin directory there is “G2D.dll” file. If you are running the examples, they will find it in the bin directory. If you build your own program, you should include the G2D.dll file in the directory of your exe file.   
In the Delphi directory, there are Pascal units that use the DLL files (from the bin) and build a Delphi Object oriented wrapper around the native C functions in the DLL. These files should be included in the uses of your project and in ti the units that use them like in the examples provided in the examples directory.

In the C directory, there are the C source of the G2D.dll. You can use them if you want to change the G2D.dll. Do that only if you are sure you know what you are doing.

#### The simplest program.

Once you installed both Gstreamer framework & G2D, you can run the simplest program (Less than 10 code lines). This program has a Gstreamer defined as a class of GstFrameWork type. In the program, we create a video test stream and show it in a window on the desktop.

You can download from examples directory or copy from here:

program PG2DExample;

{$APPTYPE CONSOLE}

uses

System.SysUtils,

G2D; // in 'G2D.pas';

{--- readme.txt ---------------------------------------------------------------

This program launches a gstreamer pipeline.

The pipeline is constructed just with two standard known plugins. a source (src) linked to a sink.

The src is "videotestsrc" that generates an endless video test signal.

The sink is "autovideosink that displays a video in a window on the screen (desktop).

------------------------------------------------------------------------------}

//main -------------------------------------------------------------------------

Var GStreamer:GstFrameWork;

begin

try

GStreamer:=GstFrameWork.Create(0,nil); //no parameters needed here

try

//launch the gstreamer pipeline

if GStreamer.LaunchSimlpePipelineAndWaitEos('videotestsrc ! autovideosink')

then writeln('GStreamer had ran until '+DateToIso(Now))

else writeln('error in the prog');

finally

if (GStreamer<>nil) and GstFrameWork.Started //check static var started cause GStreamer may not have started

then GStreamer.DisposeOf;

end;

except

on E: Exception do

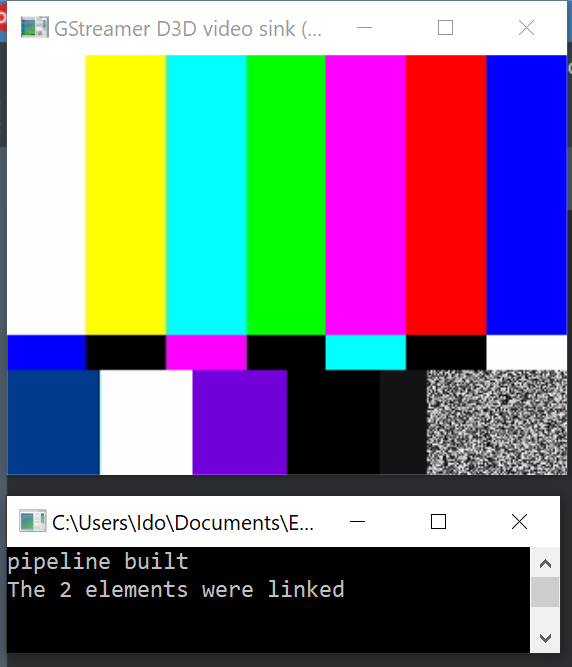
Writeln(E.ClassName, ': ', E.Message);

end;

write('press enter to exit'); readln;

end.

When running the above program you should get two windows on your desktop that look like this:



The top window (Gstreamer D3D video…) is the video output.

The bottom window (c:\users…) is a cmd line window.

If you do not get this result:

Please go back to “Installation” section and “Be Aware” section you probably did something wrong. Alternatively, your system is configured in a specific way that we did not test.

jhjhhj