## NDI GStreamer Project Notes

Contents

[NDI GStreamer Project Notes 1](#_Toc13133539)

[Overview 1](#_Toc13133540)

[Modules 2](#_Toc13133541)

[library 2](#_Toc13133542)

[sample 2](#_Toc13133543)

[ndigst 2](#_Toc13133544)

[ndielems 2](#_Toc13133545)

[ardemo 3](#_Toc13133546)

[capitogst 4](#_Toc13133547)

[Building CAPISample 4](#_Toc13133548)

[References 7](#_Toc13133549)

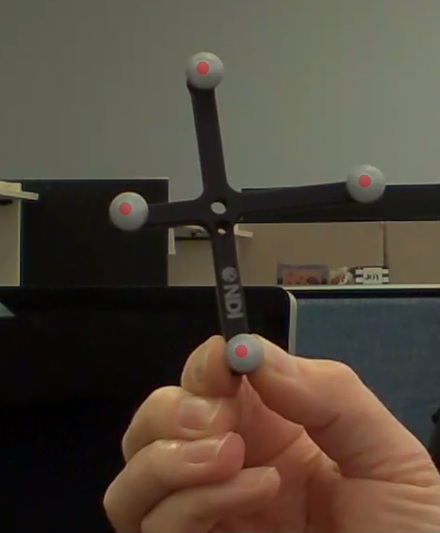
# Overview

A video camera unit is an option for the Polaris Vega. This presents opportunities for combining the tracking and video data streams for augmented reality scenarios.

GStreamer (GStreamer, n.d.) is an open source multimedia framework. It is often used for audio/video applications but can be used for any type of streaming data such as scientific signal processing ( for example, Gravitational Waves (gstlalworkshop, n.d.) (DASWG, n.d.)).

The sample code supplied shows how to leverage the GStreamer framework to create building blocks for an augmented reality application. After becoming familiar with GStreamer and the supplied sample code, it would be possible to build more building blocks (GStreamer elements).

The Combined API (CAPI) sample application (see the readme.txt in the CAPIsample root directory) is made use of for a demonstration of the VR Demo.



# Modules

The CAPISample solution contains the following directories (corresponding to Visual Studio 2017 projects in most cases):

## library

The base library defining the API.

## sample

Sample executable utilizing the library.

## ndigst

The directory containing the modules related to the GStreamer sample code, including the VS2017 solution ndigst.sln. The modules in this directory are described below.

## ndielems

The custom NDI GStreamer elements are defined in the ***ndielems*** project. The elements are:

1. **tracksrc**
   1. Source element for track data (tracking/tool/marker and other data). This element communicates with the Polaris Vega to retrieve this information and pass it down the pipeline to other elements.
2. **trackoverlay**
   1. takes as input the track data as well as the video data from the vcu. The track data is drawn as an overlay on the video data. The marker and tool positions are transformed using information from the tracksrc to correctly align with the video image.
3. **trackbin**
   1. Manages and connects all the required GStreamer elements. The trackbin has one source pin that is a video stream containing the track data overlaid on the video stream.

## ardemo

The **ardemo** project creates an executable to test the NDI GStreamer elements. The NDI elements can be built and loaded programmatically (when DO\_LOCAL\_TRACKSRC\_LOAD is defined in capisample\ndigst\ardemo\ardemobin.c, as it is by default). This way, the NDI elements are not required to be installed in the GStreamer plugin directory.

The executable can be run with the appropriate parameters, for example (change the directory path, rom file, and port as needed):

* **ardemo 169.254.176.61 c:/capisample/sroms/ 8700339.rom 554**

The parameters are:

* IP address of the PSU
* the tool definition directory
* the tool file to load
* the port for the video stream

Once the project is built and the custom NDI GStreamer elements created, the pipeline can be created and run from the command line using the GStreamer utility *gst-launch-1.0.*

The simplest way is to copy the required dlls to the GStreamer plugin and bin directory.

* Copy ***library.dll*** and ***capitogst.dll*** to: C:\gstreamer\1.0\x86\_64\bin
* Copy ***ndielems.dll*** to: C:\gstreamer\1.0\x86\_64\lib\gstreamer-1.0

Then, you can launch with the following (change the parameters as needed):

gst-launch-1.0 trackbin connect-to=169.254.176.61 rtsp-port=554 tool-location=c:/capisample/sroms/ tool-file=8700339.rom name=VegaTrackBin ! autovideosink sync=false name=VideoDisplay

Here is an example of saving the video data to a file:

gst-launch-1.0 -e trackbin connect-to=169.254.176.61 rtsp-port=554 tool-location=c:/capisample/sroms/ tool-file=8700339.rom name=VegaTrackBin ! queue ! x264enc speed-preset=ultrafast ! queue ! h264parse ! mp4mux ! filesink sync=false location=c:/work/media/nativetest7.mp4

An example saved file is nativetest.mp4 included in the doc directory.

There are three source files in the ardemo directory:

* **ardemobin.c**
* **ardemo.c**
* **demotrack.c**

By default, **ardemobin.c** is compiled in this project. This version makes use of the **targetbin** element. As an option, the ardemo.c can instead be compiled for the executable. This version creates a pipeline without using the **targetbin** element. The demotrack.c creates a very simple pipeline with only the track source element, a queue, and a fakesink. The resultant pipeline is shown in NDIVideoPipelineWithVRdemotrack.png (included in the doc directory).

## capitogst

the **capitogst** project provides a convenient interface to the CAPISample API to serialize the data in a way the is easy to use in the C development environment of GStreamer (GStreamer follows the [GObject](https://developer.gnome.org/gobject/stable/) (gobject, n.d.) programming model). The open source Google serialization library flatbuffers (flatbuffers, n.d.) is made use of in **capitogst**.

The FlatBuffer schema file is capisample\ndigst\capitogst\flatbuffer\flattooldata.fbs, if changes are made to the schema, use the flatcc compiler to regenerate the header files (the header files are in capisample\ndigst\capitogst\flatbuffer\fbheaders).

flatcc -a -o fbheaders flattooldata.fbs

For this project, flatcc version: 0.6.0-pre was used. The binaries and generated libraries are part of the project and can be found in capisample\ndigst\external\flatcc.

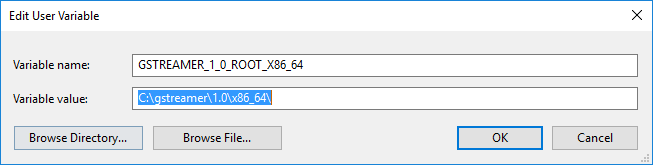
# Building CAPISample

Prior to building the GStreamer related components described above, GStreamer needs to be installed. The windows installation notes for GStreamer can be found on the GStreamer website (GStreamer Windows Install, n.d.). You will need to install both the GStreamer runtime, and the development package (The are two separate installations). The project was developed with GStreamer version 1.14.4. An introduction to GStreamer plugin development can be found on the GStreamer website (plugin-development, n.d.). Also, a pdf version of the GStreamer Application Development Manual is included in the doc directory.

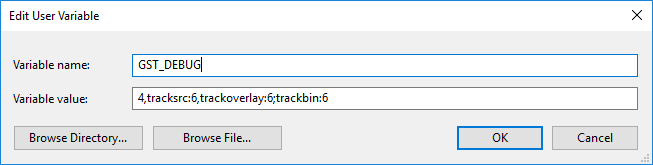
Once installed, verify that you can, for example, at the command prompt successfully run:

gst-inspect-1.0 rtspsrc

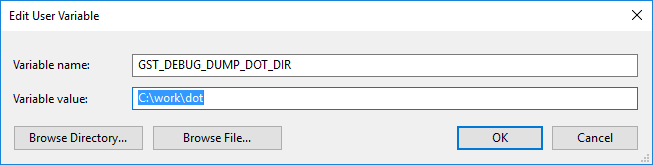
Also verify that the environment variable is defined (the visual studio projects refer to this environment variable for include files, library locations and props files):



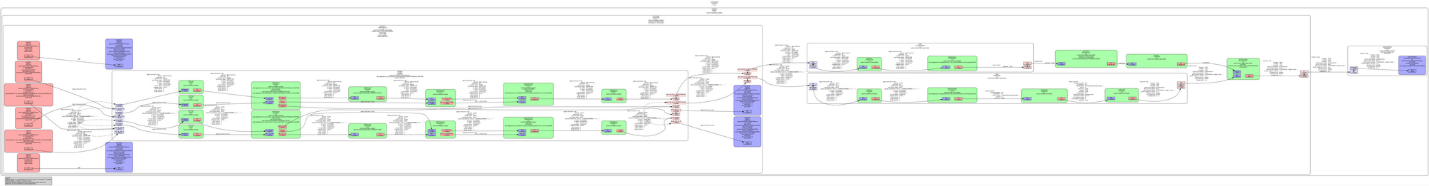
Defining the following environment variable is helpful for getting more specific debug information from the NDI elements:



This environment variable is helpful for examining the pipeline created.



For example, this is the output after launching the ardemo (click on image for larger view, or see NDIVideoPipelineWithBin.png in the doc directory).

[](NDIVideoPipelineWithBin.png)

The pipeline makes extensive use of existing GStreamer elements. The rtspsrc takes up most of the real estate of the image. It is a bin element that makes use of many other elements.

The custom NDI elements are (as described earlier in this document) are also shown in the image. The trackbin element connects and manages most of the elements shown, except the autovideosink used to show the resulting video stream.

For building in Windows, solutions and project files for VS2017 are supplied. The Windows SDK version we used for development is 10.0.16299.0 . Depending on when you installed Visual Studio 2017, you may have a different SDK. When building, if the SDK is not installed, you will see the error:

MSB8036 The Windows SDK version 10.0.16299.0 was not found. Install the required version of Windows SDK or change the SDK version in the project property pages or by right-clicking the solution and selecting "Retarget solution".

If you wish to use the same SDK, run the Visual 2017 installer, click modify, select the “Individual components” tab, and select the needed SDK (in the SDKs, libraries, and frameworks section, as shown in the image below):

A screenshot of a cell phone

Description generated with very high confidence

# References

(n.d.). Retrieved from GStreamer: https://gstreamer.freedesktop.org/documentation/index.html?gi-language=c

*DASWG*. (n.d.). Retrieved from https://wiki.ligo.org/Computing/DASWG/GstLAL

*flatbuffers*. (n.d.). Retrieved from https://google.github.io/flatbuffers/

*gobject*. (n.d.). Retrieved from https://developer.gnome.org/gobject/stable/

*gstlalworkshop*. (n.d.). Retrieved from http://www.resceu.s.u-tokyo.ac.jp/~kipp/gstlalworkshop/intro\_and\_setup.html

*GStreamer Windows Install*. (n.d.). Retrieved from https://gstreamer.freedesktop.org/documentation/installing/on-windows.html?gi-language=c

*plugin-development*. (n.d.). Retrieved from https://gstreamer.freedesktop.org/documentation/plugin-development/index.html?gi-language=c

*Windows*. (n.d.). Retrieved from https://gstreamer.freedesktop.org/documentation/installing/on-windows.html?gi-language=c