Josh’s GStreamer Color Video Filter: Analysis

Overview

Josh’s GStreamer Color Video Filter is a self-building GStreamer plugin that adds a video filter capable of modifying RGB values of streaming video. It has a Meson build environment that compiles all code and adds compiled files directly to a computer’s GStreamer libraries, seamlessly integrating the code into a user’s GStreamer elements.

Github repository: https://github.com/sjoshuserful/Video-Filter

Github Readme

readme.md

Josh's GStreamer Color Video Filter

This is code for a GStreamer element "customfilter" that can take in any RGB video and filter out any of the RGB channels.

Key:

filter-mode == 0: No change

filter-mode == 1: All red light filtered from video

filter-mode == 2: All green light filtered from video

filter-mode == 3: All blue light filtered from video

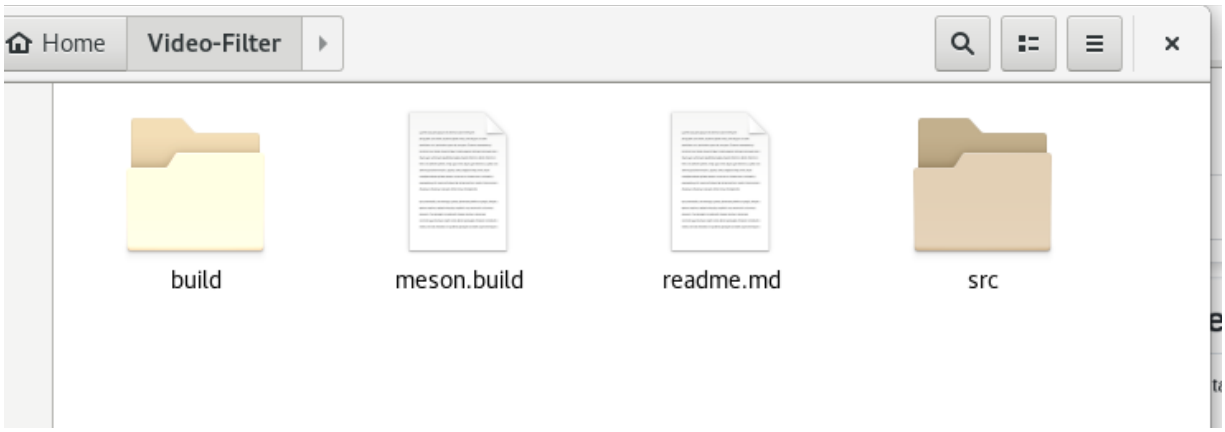
Installation

Josh's Color Video Filter can be easily built and integrated into the GStreamer libraries as it has a custom Meson build environment. Simply complete the following steps:

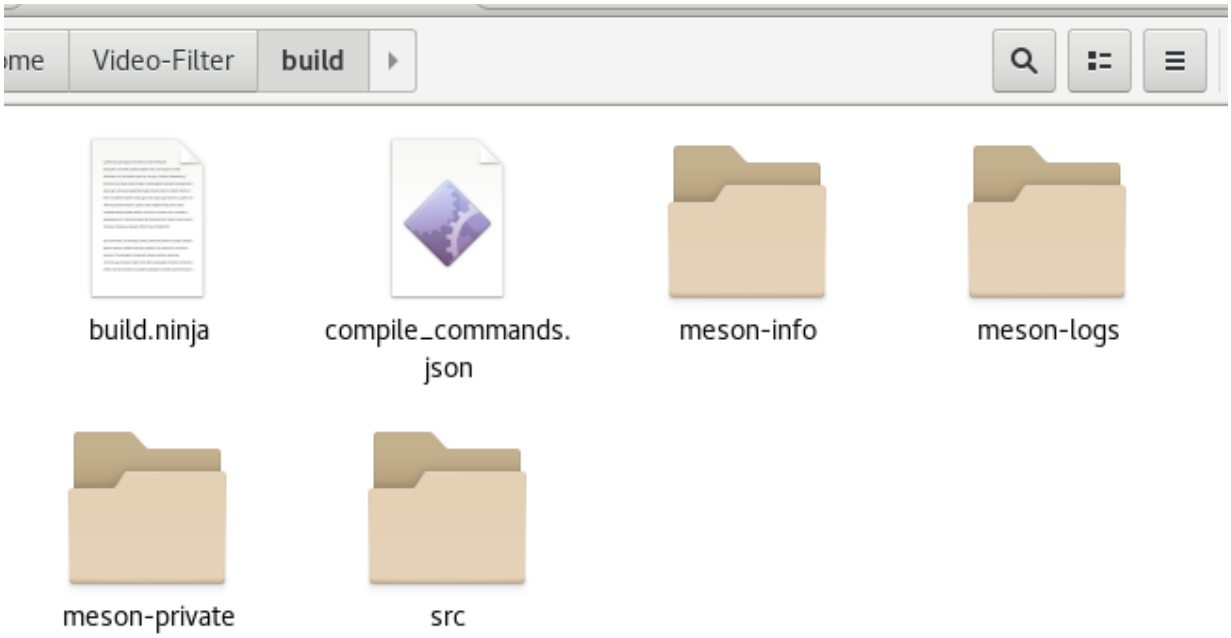
1. Clone this repository
2. Navigate to the directory in terminal, then navigate to /Video-Filter/build/
3. Run: $ ninja
4. Run: $ sudo ninja install

"customfilter" will now be recognized (provided you use /usr/lib64/gstreamer-1.0/ as your GStreamer libraries directory).

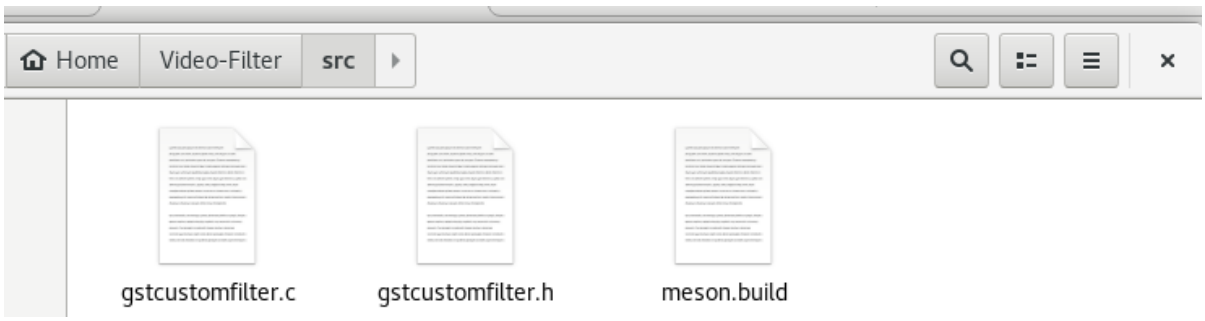
Build Environment



Project Directory



Build Directory



Source Directory

The main project directory, Video-Filter, contains a source directory, a build directory, a readme, and a meson.build. Code is edited in the source directory, and build commands are run in the build directory, creating a clean, organized environment.

Meson

Video-Filter contains two meson.build files:

```
project('gst-customfilter', 'c',
  version : '0.0.1',
  license : ['LGPL'],
  meson_version : '>= 0.52.0',
  default_options : [
    'warning_level=2',
    'buildtype=debug',
    'c_std=gnu99'
  ]
)

cc = meson.get_compiler('c')

gst_api_version = '1.0'
glib = dependency('glib-2.0')
dep_gst  = \
    [ dependency('gstreamer-1.0',              include_type: 'system')
    , dependency('gstreamer-app-1.0',          include_type: 'system')
    , dependency('gstreamer-base-1.0',         include_type: 'system')
    , dependency('gstreamer-pbutils-1.0',      include_type: 'system')
    , dependency('gstreamer-plugins-base-1.0', include_type: 'system')
    , dependency('gstreamer-rtsp-1.0',         include_type: 'system')
    , dependency('gstreamer-rtsp-server-1.0',  include_type: 'system')
    , dependency('gstreamer-video-1.0',        include_type: 'system')
    ]

#data_install_dir = get_option('datadir') / meson.project_name()

inc = include_directories('.')

subdir('src')
```

This is the main meson.build file in the /Video-Filter/ directory, and it is read first. After initializing the project, many dependencies are listed and added to the compiler. These libraries make it possible for this same build environment to be utilized for other types of elements in the future.

```
gstcustomfilter_sources = [
  'gstcustomfilter.c'
]
gstcustomfilter_deps = [
  glib,
  dep_gst
]

if host_machine.system() == 'linux'
  gstcustomfilter_deps += cc.find_library('dl', required : true)
endif

gstcustomfilter = library('gstcustomfilter',
  gstcustomfilter_sources,
  dependencies : gstcustomfilter_deps,
  c_args : [
    '-DGST_USE_UNSTABLE_API'
  ],
  include_directories : inc,
  install : true,
  install_dir : '/usr/lib64/gstreamer-1.0'
)

```

This is the meson.build file inside the /Video-Filter/src/ directory, and it is read second. This is what creates the library program files with the included directories. Install_dir installs these compiled program files into the gstreamer libraries directory and makes it instantly accessible to a user.

gstcustomfilter.c: Property variables and Caps

```
63  enum
64  {
65    PROP_0,
66    PROP_FILTER_MODE
67  };
68
69  /* pad templates */
70
71  /* FIXME: add/remove formats you can handle */
72  #define VIDEO_SRC_CAPS \
73      GST_VIDEO_CAPS_MAKE("{RGB}")
74
75  /* FIXME: add/remove formats you can handle */
76  #define VIDEO_SINK_CAPS \
77      GST_VIDEO_CAPS_MAKE("{RGB}")
78
```

Initializing PROP_FILTER_MODE to represent the filter-mode property responsible for changing the element filter color. Caps are set to RGB.

Gstcustomfilter.c: customfilter Class Initialization

```
86  static void
87  gst_customfilter_class_init (GstCustomfilterClass * klass)
88  {
89    GObjectClass *gobject_class = G_OBJECT_CLASS (klass);
90    GstBaseTransformClass *base_transform_class = GST_BASE_TRANSFORM_CLASS (klass);
91    GstVideoFilterClass *video_filter_class = GST_VIDEO_FILTER_CLASS (klass);
92
93    /* Setting up pads and setting metadata should be moved to
94       base_class_init if you intend to subclass this class. */
95    gst_element_class_add_pad_template (GST_ELEMENT_CLASS(klass),
96        gst_pad_template_new ("src", GST_PAD_SRC, GST_PAD_ALWAYS,
97          gst_caps_from_string (VIDEO_SRC_CAPS)));
98    gst_element_class_add_pad_template (GST_ELEMENT_CLASS(klass),
99        gst_pad_template_new ("sink", GST_PAD_SINK, GST_PAD_ALWAYS,
100         gst_caps_from_string (VIDEO_SINK_CAPS)));
101
102   gst_element_class_set_static_metadata (GST_ELEMENT_CLASS(klass),
103       "Josh's Video Filter", "Filter/Effect/Video", "Filter out certain color from streamed video",
104       "Josh Strand josh.strand@userful.com");
105
106   gobject_class->set_property = gst_customfilter_set_property;
107   gobject_class->get_property = gst_customfilter_get_property;
108   gobject_class->dispose = gst_customfilter_dispose;
109   gobject_class->finalize = gst_customfilter_finalize;
110   base_transform_class->start = GST_DEBUG_FUNCPTR (gst_customfilter_start);
111   base_transform_class->stop = GST_DEBUG_FUNCPTR (gst_customfilter_stop);
112   video_filter_class->set_info = GST_DEBUG_FUNCPTR (gst_customfilter_set_info);
113   video_filter_class->transform_frame = GST_DEBUG_FUNCPTR (gst_customfilter_transform_frame);
114
115   g_object_class_install_property (gobject_class, PROP_FILTER_MODE,
116       g_param_spec_uint ("filter-mode", "Sets RGB filter",
117           "Filters R,G,B. filter-mode=0: no filter, filter-mode=1: filter red, \
118           filter-mode=2: filter green, filter-mode=3: filter blue", 0,
119           3, 0, G_PARAM_READWRITE));
120
121 }
122
```

Class initialization function for a customfilter class. Initializes source and sink pads, sets data for gst-inspect, provides a key to the user on how to configure the color filter, and sets members of the class to functions defined in the library.

Gstcustomfilter.c: Filter Initialization Function, Set_Property, and Get_Property

```c
static void
gst_customfilter_init (GstCustomfilter *customfilter)
{
    GST_DEBUG_OBJECT (customfilter, "Initializing the element");
    customfilter->filtermode = 0;
}

void
gst_customfilter_set_property (GObject * object, guint property_id,
    const GValue * value, GParamSpec * pspec)
{
  GstCustomfilter *customfilter = GST_CUSTOMFILTER (object);

  // Set property according to property ID
  switch(property_id) {
      case PROP_FILTER_MODE:
        customfilter->filtermode = g_value_get_uint(value);
        break;
      default:
        G_OBJECT_WARN_INVALID_PROPERTY_ID (object, property_id, pspec);
        break;
  }

  GST_DEBUG_OBJECT (customfilter, "set_property");

}

void
gst_customfilter_get_property (GObject * object, guint property_id,
    GValue * value, GParamSpec * pspec)
{
  GstCustomfilter *customfilter = GST_CUSTOMFILTER (object);

  GST_DEBUG_OBJECT (customfilter, "get_property");

  switch (property_id) {
    case PROP_FILTER_MODE:
        g_value_set_uint(value, customfilter->filtermode);
        break;
    default:
        G_OBJECT_WARN_INVALID_PROPERTY_ID (object, property_id, pspec);
        break;
  }
}
```

Gst_customfilter_init initializes the element with default filter-mode = 0.
Gst_customfilter_set_property takes the filter-mode number from the user input and saves it to member “filtermode” of a GstCustomFilter object.
Gst_customfilter_get_property sets the “value” Gvalue to the filtermode.

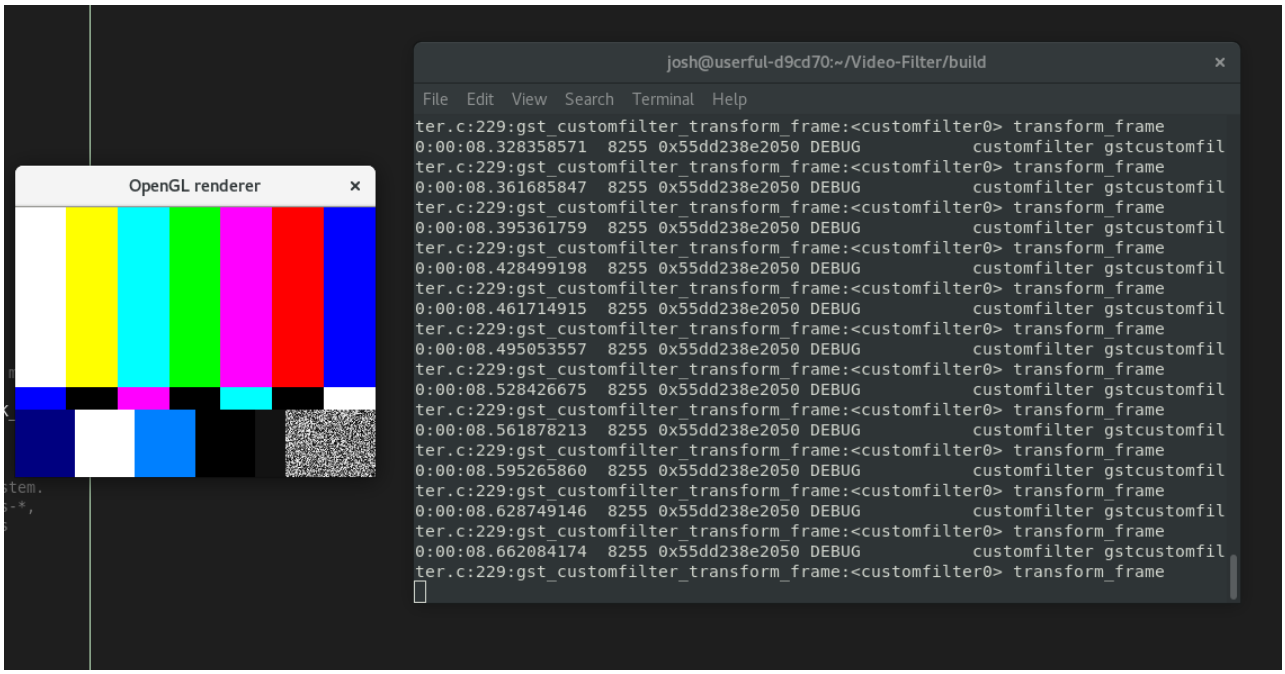
Gstcustomfilter.c: Transform Function

```c
/* transform */
static GstFlowReturn
gst_customfilter_transform_frame (GstVideoFilter * filter, GstVideoFrame * inframe,
    GstVideoFrame * outframe)
{
  GstCustomfilter *customfilter = GST_CUSTOMFILTER (filter);
  GST_DEBUG_OBJECT (customfilter, "transform_frame");

  gst_video_frame_copy(outframe, inframe);

  guint8 *pixels = GST_VIDEO_FRAME_PLANE_DATA (outframe, 0);
  guint stride = GST_VIDEO_FRAME_PLANE_STRIDE (outframe, 0);
  guint pixel_stride = GST_VIDEO_FRAME_COMP_PSTRIDE (outframe, 0);

  guint height = GST_VIDEO_FRAME_HEIGHT(outframe);
  guint width = GST_VIDEO_FRAME_WIDTH(outframe);

     if (PROP_FILTER_MODE) {
        for (guint h = 0; h < height; ++h) {
            for (guint w = 0; w < width; ++w) {
                guint8 *pixel = pixels + h * stride + w * pixel_stride;
                // filter-mode == 1: Filter the red
                // filter-mode == 2: Filter the green
                // filter-mode == 3: Filter the blue
                //red
                if (customfilter->filtermode == 1) {
                    memset(pixel, 0, 1);
                }
                else {memset(pixel, *pixel, 1);}
                pixel++;

                // green
                if (customfilter->filtermode == 2) {
                    memset(pixel, 0, 1);
                }
                else {memset(pixel, *pixel, 1);}
                pixel++;

                //blue
                if (customfilter->filtermode == 3) {
                    memset(pixel, 0, 1);
                }
                else {memset(pixel, *pixel, 1);}
            }
        }
    }
   //}

  return GST_FLOW_OK;
}
```

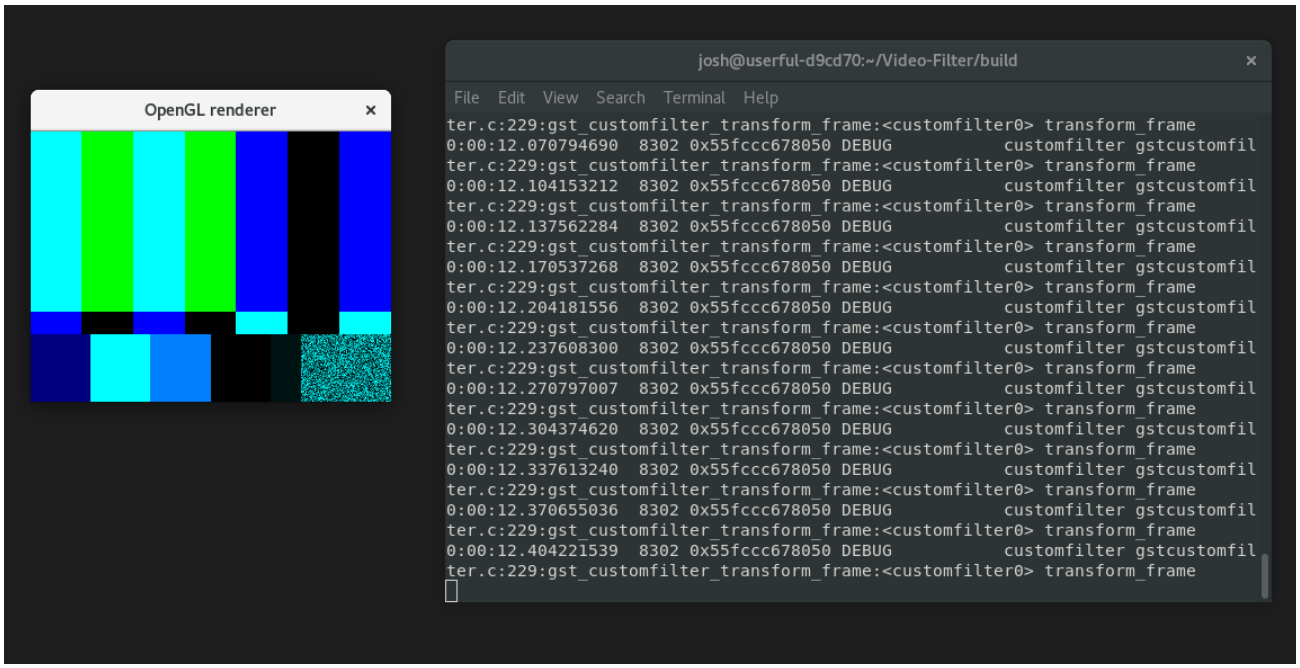
Gst_customfilter_transform_frame runs each frame. While it runs it takes in the current frame, copies it to the outgoing frame, obtains pointers to the first pixels in the fram, iterates over each pixel, and sets RGB values according to how the user decided. The

traversal of the frame is accomplished by nested for loops that memset pixels and increment pixel pointers to write to each RGB channel.

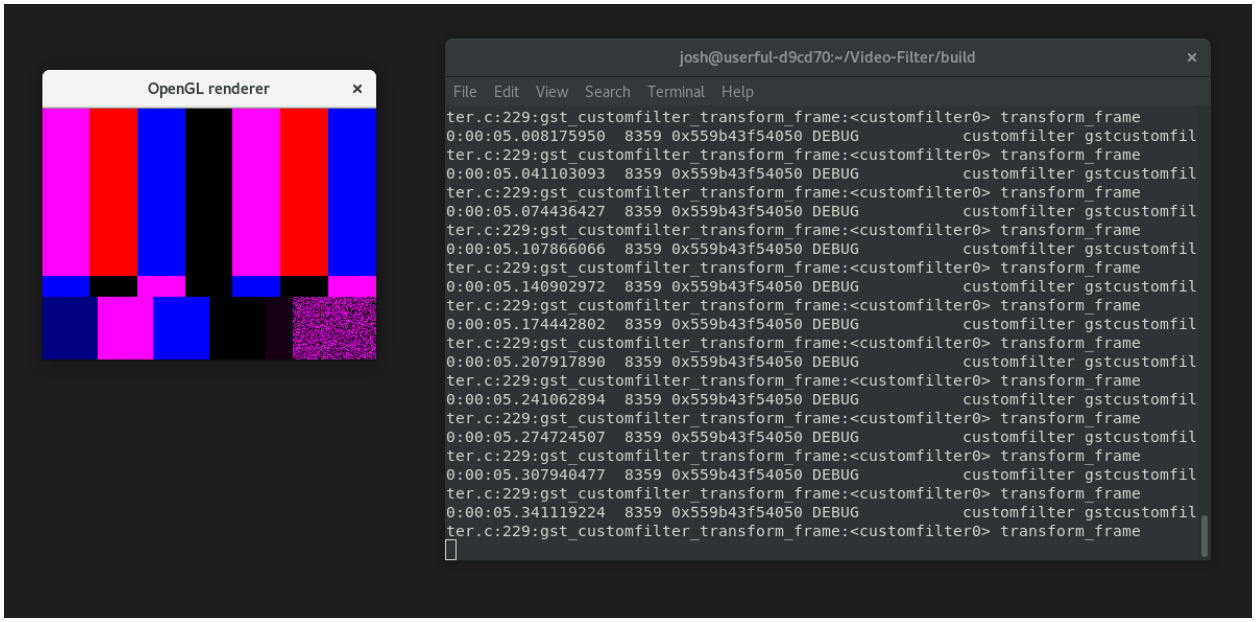
Demo



No Filter



Red Filter



Green Filter

Blue Filter