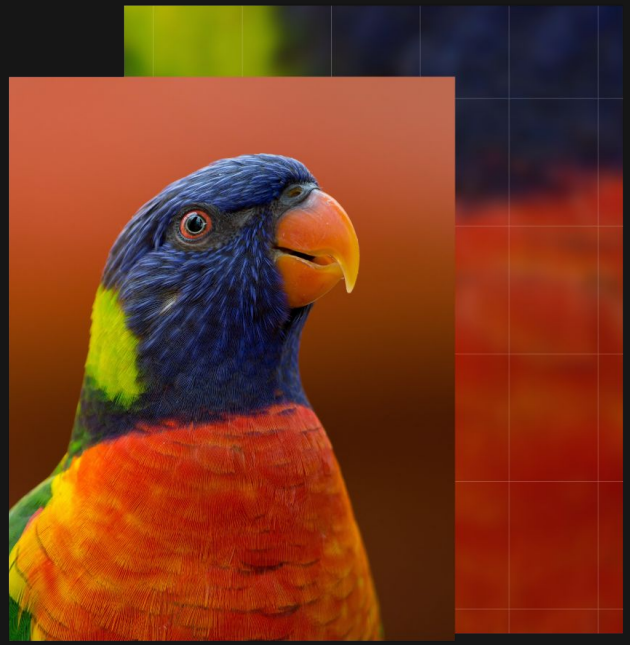
Patching 3rd-party plugin in runtime

Using one curious side effect of the Glib Dynamic Type System.



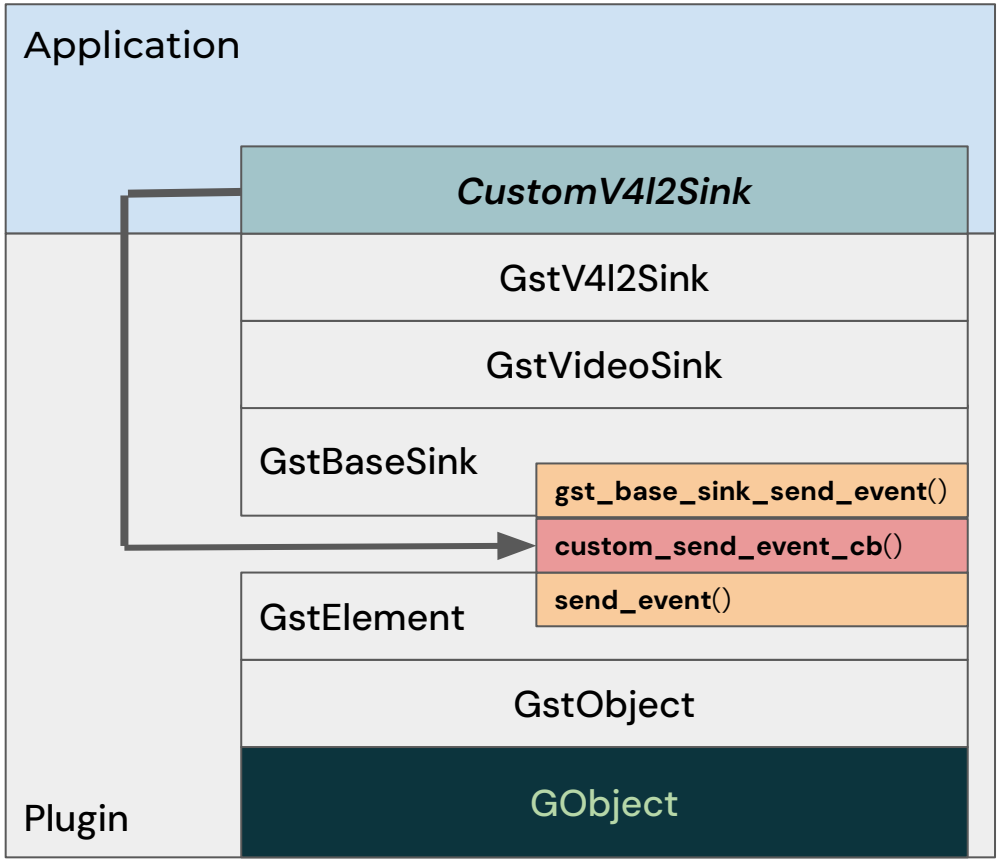
fluendo

GStreamer Conference 2023, Alexander Slobodeniuk

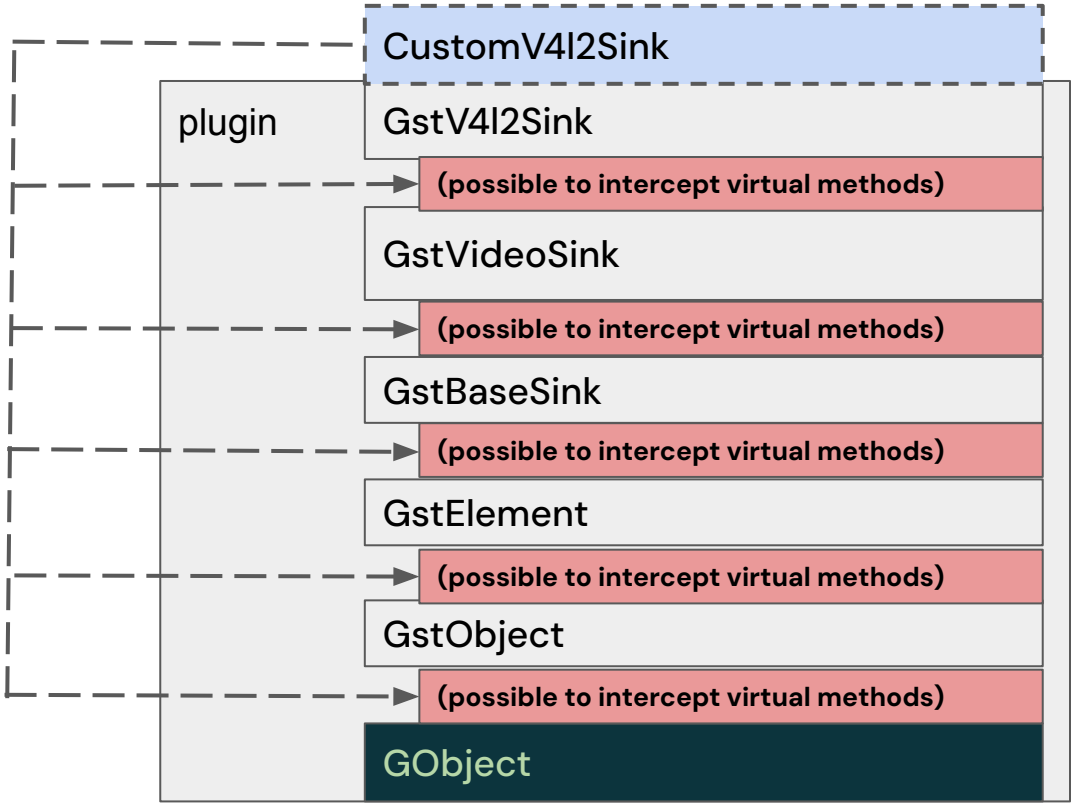
Index

Overview

The goal of this presentation is to demonstrate some aspects of how Glib Dynamic Type System works, and how GStreamer framework manages to register its elements.



Custom code is a mixin, that is applied to a new class registered using any GStreamer element as a base class.



Registration of a new GstElement

```
static gboolean
plugin_init (GstPlugin * plugin)
{
...
  ret |= GST_ELEMENT_REGISTER (v4l2sink, plugin);
...
 }
```



```
gst_element_register (plugin, "v4l2sink", GST_RANK_PRIMARY, GST_TYPE_V4L2SINK);
```

gstv4l2sink.h

```
#define GST_TYPE_V4L2SINK \
  (gst_v4l2sink_get_type())
```

gstv4l2sink.c

```
G_DEFINE_TYPE_WITH_CODE (GstV4l2Sink, gst_v4l2sink, GST_TYPE_VIDEO_SINK,
    G_IMPLEMENT_INTERFACE (GST_TYPE_TUNER, gst_v4l2sink_tuner_interface_init);
    G_IMPLEMENT_INTERFACE (GST_TYPE_COLOR_BALANCE,
        gst_v4l2sink_color_balance_interface_init);
    G_IMPLEMENT_INTERFACE (GST_TYPE_VIDEO_ORIENTATION,
        gst_v4l2sink_video_orientation_interface_init));
```

G_DEFINE_TYPE / G_DEFINE_TYPE_WITH_CODE

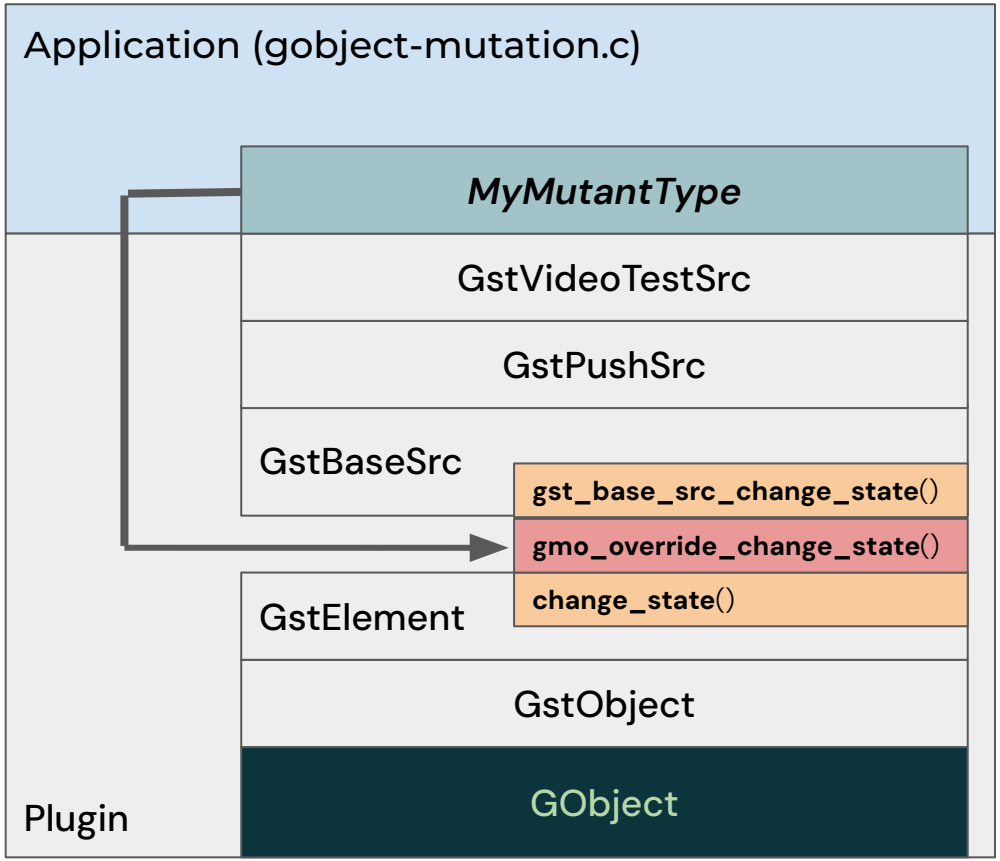
```
GType
gst_v4l2sink_get_type (void)
{
 static gsize v4l2sink_type = 0;

  if (g_once_init_enter (&v4l2sink_type)) {
    static const GTypeInfo base_sink_info = {
     ...
    };
    _type = g_type_register_static (GST_TYPE_VIDEO_SINK,
       "GstV4l2Sink", &v4l2sink_info, 0);

... (install interfaces)...

    g_once_init_leave (&v4l2sink_type, _type);
  }

  return v4l2sink_type;
}
```

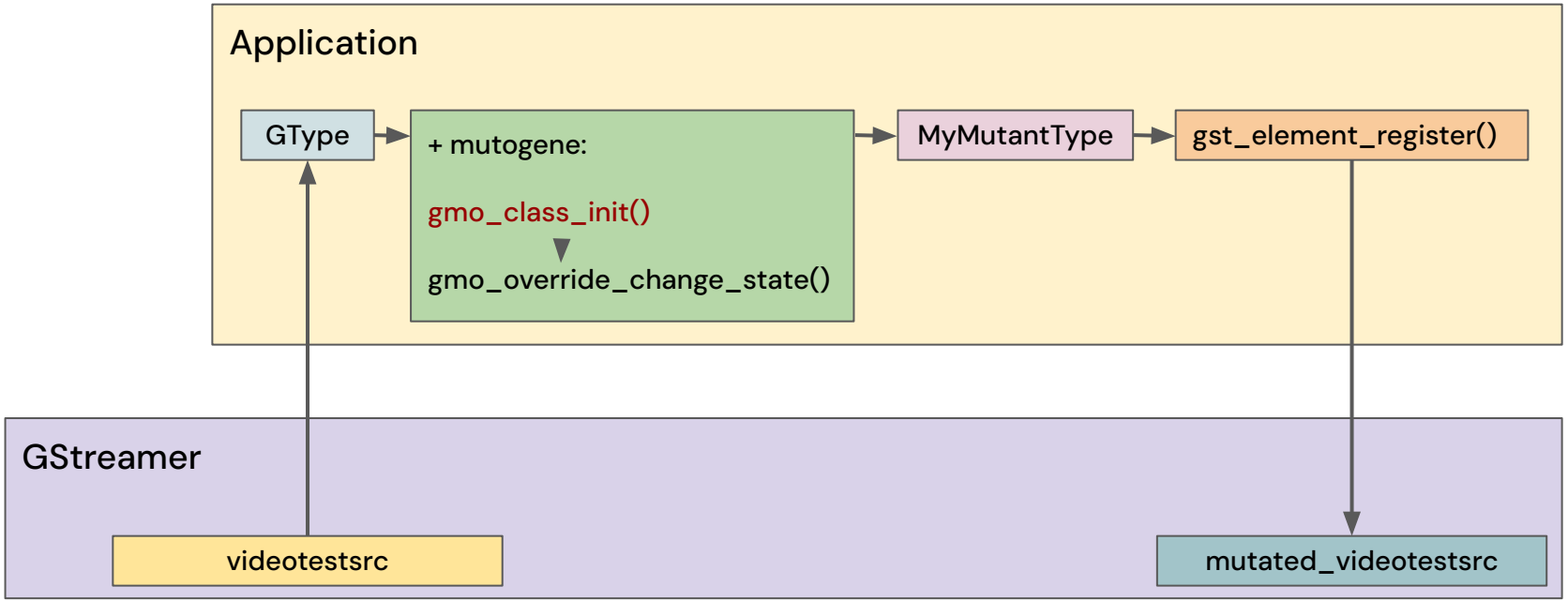
struct GTypeInfo

```
struct GTypeInfo {
  /* interface types, classed types, instantiated types */
  guint16             class_size;

  GBaseInitFunc       base_init;
  GBaseFinalizeFunc   base_finalize;

  /* interface types, classed types, instantiated types */
  GClassInitFunc      class_init;
  GClassFinalizeFunc  class_finalize;
  gconstpointer       class_data;

  /* instantiated types */
  guint16             instance_size;
  guint16             n_preallocs;
  GInstanceInitFunc   instance_init;

  /* value handling */
  const GTypeValueTable *value_table;
};
```

Walking through a code example

Code example can be found here (200 lines of code).




Application
GType
+ mutogene:
gmo_class_init()
gmo_override_change_state()
MyMutantType
gst_element_register()
GStreamer
videotestsrc
mutated_videotestsrc

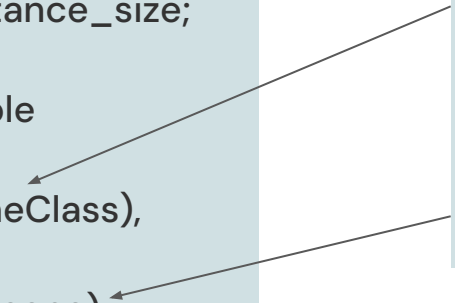
Getting the source type

```
GType dna_type;
...

{                       /* Get the DNA type */
  GstElementFactory *fac;

  fac = GST_ELEMENT_FACTORY (gst_plugin_feature_load (GST_PLUGIN_FEATURE
        (gst_element_factory_find ("videotestsrc"))));
  dna_type = gst_element_factory_get_element_type (fac);
  gst_object_unref (fac);
}
```

Registering the mutant type

```
g_type_query (dna, &dna_info);
 /* set offsets */
 gmo_dna_class_offset = dna_info.class_size;
 gmo_dna_instance_offset = dna_info.instance_size;

 return (ret = g_type_register_static_simple
    (dna, "MyMutantType",
        dna_info.class_size + sizeof (MutogeneClass),
        gmo_class_init,
        dna_info.instance_size + sizeof (Mutogene),
        gmo_init,
        0));
```

```
typedef struct
{
  int abc;
} MutogeneClass;

typedef struct
{
  guint num;
} Mutogene;
```



Intercepting a virtual function

```
static void
gmo_class_init (gpointer dna, gpointer class_data)
{
  MutogeneClass *klass;
  GstElementClass *element_class = (GstElementClass *) dna;

  parent_class = g_type_class_peek_parent (dna);

  klass = gmo_get_class (dna);
  klass->abc = 123;

  element_class->change_state = gmo_override_change_state;
}
```

```
static GstStateChangeReturn
gmo_override_change_state (GstElement * element,
GstStateChange transition)
{
  Mutogene *mut = gmo_get_mutogene (element);

  g_message ("Changing state to %s. Will count it as number %d",
      gst_element_state_get_name
(GST_STATE_TRANSITION_NEXT (transition)),
      ++mut->num);

  return parent_class->change_state (element, transition);
}
```

Chain up to parent class (GstVideoTestSrcClass)

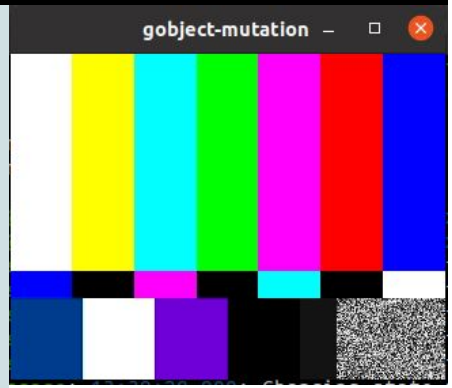
Registering a new GStreamer element

```
/* Register our mutant as GstElement */
 if (!gst_element_register (NULL, "mutated_videotestsrc", 999,
        mutated_dna_type)) {
   g_error ("Couldn't register mutant element");
 }

 pipe = gst_parse_launch ("mutated_videotestsrc ! autovideosink", NULL);
 g_assert (NULL != pipe);
 gst_element_set_state (pipe, GST_STATE_PLAYING);
```

Running the example

```
> cc gobject-mutation.c $(pkg-config --cflags --libs gstreamer-1.0) -o gobject-mutation
> ./gobject-mutation
** Message: 13:46:09.380: Hello from MyMutantType, the child of GstVideoTestSrc
** Message: 13:46:09.394: Changing state to READY. Will count it as number 1
** Message: 13:46:09.394: Changing state to PAUSED. Will count it as number 2
** Message: 13:46:09.408: Changing state to PLAYING. Will count it as number 3
** Message: 13:46:29.395: Changing state to PAUSED. Will count it as number 4
** Message: 13:46:29.396: Changing state to READY. Will count it as number 5
** Message: 13:46:29.430: Changing state to NULL. Will count it as number 6
>
```

```c
static void gmo_init (GTypeInstance * dna, gpointer klass)
{
  Mutogene *mut;

  mut = gmo_get_mutogene (dna);
  mut->num = 0;

  g_message ("Hello from %s, the child of %s", G_OBJECT_CLASS_NAME (klass),
      G_OBJECT_CLASS_NAME (parent_class));
}
```



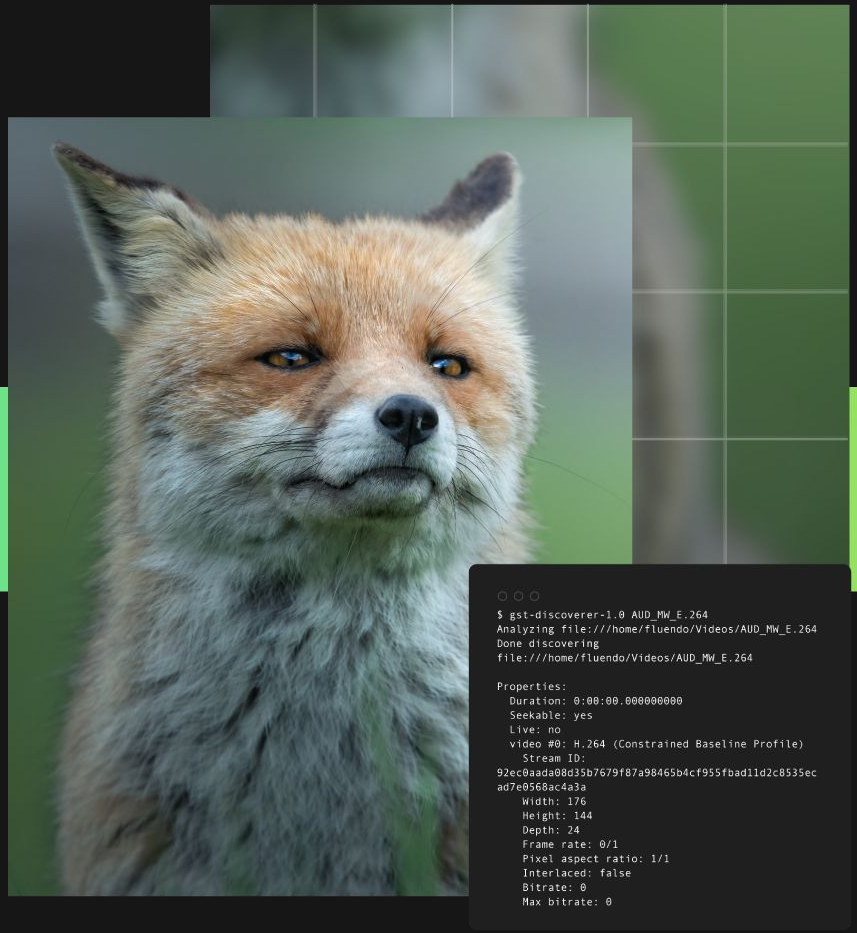
Possible use cases

- GStreamer already has pad probes that cover most of the need, and also tracers
- Any virtual function of an available base class can be intercepted
- The implementation of the “patch” (mutogene) is agnostic to it’s parent.
- The method seems to be stable and clear, but still looks like a hack

Examples that appear in mind:

- add “preroll” or “render” signals to any sink
- intercept handling an upstream event on any sink
- add “fill” signal to any source
- intercept metadata filtering on encoders or decoders
- intercept “provide_clock” of GstElement

Questions



```
$ gst-discoverer-1.0 AUD_MW_E.264
Analyzing file:///home/fluendo/Videos/AUD_MW_E.264
Done discovering
file:///home/fluendo/Videos/AUD_MW_E.264

Properties:
  Duration: 0:00:00.000000000
  Seekable: yes
  Live: no
  video #0: H.264 (Constrained Baseline Profile)
    Stream ID:
92ec0aada08d35b7679f87a98465b4cf955fbad11d2c8535ec
ad7e0568ac4a3a
    Width: 176
    Height: 144
    Depth: 24
    Frame rate: 0/1
    Pixel aspect ratio: 1/1
    Interlaced: false
    Bitrate: 0
    Max bitrate: 0
```

Thank you!