gstreamermm

C++ way of doing GStreamer-based applications

Marcin Kolny
marcin.kolny@gmail.com

GStreamer Conference 2016, Berlin

October 10, 2016

Agenda

- Foundations & Core classes
- Writing C++ plugins
- The Future

What is gstreamermm?

- C++ wrapper for GStreamer
- provides convenient API
- assures type safety

Automatic memory management

```
1  {
2     Glib::RefPtr<Gst::Bin> bin = Gst::Bin::create();
3     // RefCount: 1
4     {
5        Glib::RefPtr<Gst::Bin> new_bin = bin;
6        // RefCount: 2
7     }
8     // RefCount: 1
9  }
10 // bin has been destroyed
```

Calling GStreamer C API

```
1  Glib::RefPtr<Gst::Pipeline> pipeline =
2                                      Gst::Pipeline::create();
3
4  // access to underlying C-object
5  GstPipeline* c_pipeline = pipeline->gobj();
6
7  // either:
8  gst_element_set_state(GST_ELEMENT(c_pipeline), GST_STATE_PLAYING);
9
10 // or:
11 pipeline->set_state(Gst::STATE_PLAYING);
```

Signals

Glib::SignalProxy - strongly typed wrapper for GObject signals

```
1  decodebin->signal_pad_added().connect(
2  [] (const RefPtr<Pad>& pad) {
3    RefPtr<Caps> caps = pad->get_current_caps();
4
5    string media_type =
6        caps->get_structure(0).get_name();
7
8    if (regex_search(media_type, regex("^video"))) {
9      pad->link(video_sink->get_static_pad("sink"));
10   } else {
11     cerr << "Unsupported media type" << endl;
12   }
13 });
```

Gst::Structure - C API

```
1  // 1 - missing NULL at the end
2  GstStructure *s0 = gst_structure_new("foo",
3       "field1", G_TYPE_DOUBLE, 3.0,
4       "field2", G_TYPE_INT, 66);
5
6  // 2 - missing field types
7  GstStructure *s1 = gst_structure_new("foo",
8       "field1", 3.0,
9       "field2", 66,
10      NULL);
11
12 // 3 - invalid type of value
13 GstStructure *s2 = gst_structure_new("foo",
14      "field1", G_TYPE_DOUBLE, 3,
15      "field2", G_TYPE_INT, 66,
16      NULL);
```

Gst::Structure - C API

```
1  // 1 - missing NULL at the end
2  GstStructure *s0 = gst_structure_new("foo",
3       "field1", G_TYPE_DOUBLE, 3.0,
4       "field2", G_TYPE_INT, 66);
5
6  // 2 - missing field types
7  GstStructure *s1 = gst_structure_new("foo",
8       "field1", 3.0,
9       "field2", 66,
10      NULL);
11
12 // 3 - invalid type of value
13 GstStructure *s2 = gst_structure_new("foo",
14      "field1", G_TYPE_DOUBLE, 3,
15      "field2", G_TYPE_INT, 66,
16      NULL);
```

Segmentation fault

Gst::Structure - C++ API

```
1 Gst::Structure structure("struct-name",
2                          "field1", 3,
3                          "field2", 66);
4
5 // Invalid:
6 Gst::Structure structure("struct-name",
7                          "field1", 3,
8                          "field2");
```

Gst::Structure - C++ API

```
Gst::Structure structure("struct-name",
                         "field1", 3,
                         "field2", 66);

// Invalid:
Gst::Structure structure("struct-name",
                         "field1", 3,
                         "field2");
```

```
# Compilation error:
/usr/local/include/gstreamermm-1.0/gstreamermm/
structure.h:523:3: error: no matching function
for call to   Gst::Structure::set_fields(const char*&)
```

Gst::Structure - custom data type

```
1  class TextHolder
2  {
3    std::string text;
4
5  public:
6    TextHolder() {}
7    TextHolder(const std::string& text)
8      : text(text) {}
9
10   std::string get_text() const  { return text; }
11 };
12
13 Gst::Structure structure("struct-name");
14
15 // automatic TextHolder type registration
16 structure.set_field("text-holder", TextHolder("Hello Berlin!"));
17
18 TextHolder holder;
19 if (structure.get_field("holder", holder)) {
20   std::cout << holder.get_text() << std::endl;
21 }
```

Gst::Message

- Base class: Gst::Message
- Specialization classes: Gst::MessageError, Gst::MessageEos etc.
- Parse methods: T parse_*(), void parse(T1& v1, T2& v2, ...)
- The same concept for Gst::Event and Gst::Query

```
1  void process_message(const RefPtr<Message>& msg)
2  {
3    switch (msg->get_message_type()) {
4      case Gst::MESSAGE_PROGRESS:
5        RefPtr<MessageProgress> msg_prg = msg_prg.cast_static(msg);
6        std::cout << msg_prog->parse_code() << std::endl;
7        break;
8
9      // other cases...
10   }
11 }
```

Plugins - declaration

```
1 class MMElement : public Gst::Element
2 {
3 public:
4   explicit MMElement(GstElement *gobj);
5
6   static void class_init(
7         Gst::ElementClass<MMElement> *klass);
8 };
```

Plugins - implementation

```
1  void MMElement::class_init(
2       Gst::ElementClass<MMElement> *klass)
3  {
4    klass->set_metadata(
5              "Plugin",
6              "MM/Examples",
7              "Simple MM element",
8              "Marcin Kolny <marcin.kolny@gmail.com>");
9  }
10
11 MMElement::MMElement(GstElement *gobj)
12   : Glib::ObjectBase(typeid(MMElement)),
13   Gst::Element(gobj)
14 {
15 }
```

Plugins - initialization

```
1 static gboolean plugin_init (GstPlugin * plugin)
2 {
3   return gst_element_register(plugin,
4       "mmelement",
5       GST_RANK_NONE,
6       Gst::register_mm_type<MMElement>("MMElement"));
7 }
```

Plugins - pads

```
1  void MMElement::class_init(Gst::ElementClass<MMElement> *klass)
2  {
3    auto sink_template = Gst::PadTemplate::create(
4                    "sink",
5                    Gst::PAD_SINK,
6                    Gst::PAD_ALWAYS,
7                    Gst::Caps::create_any());
8    klass->add_pad_template(sink_template);
9    // auto src_template = ...
10   // klass->set_metadata(...);
11 }
12
13 MMElement::MMElement(GstElement *gobj)
14   : Glib::ObjectBase(typeid(MMElement)),
15   Gst::Element(gobj)
16 {
17   add_pad(sinkpad = Gst::Pad::create(get_pad_template("sink"), "sink"));
18   GST_PAD_SET_PROXY_CAPS (sinkpad->gobj());
19
20   add_pad(srcpad = Gst::Pad::create(get_pad_template("src"), "src"));
21
22   sinkpad->set_chain_function(sigc::mem_fun( *this, &MMElement::chain));
23 }
24
25 Gst::FlowReturn MMElement::chain(const Glib::RefPtr<Gst::Pad>& pad,
26                                  Glib::RefPtr<Gst::Buffer>& buf)
27 {
28   std::cout << "Buffer size: " << buf->get_size() << std::endl;
29   return srcpad->push(std::move(buf));
30 }
```

Plugins - Gst::BaseTransform

```
1  class MMElement :public Gst::BaseTransform
2  {
3  public:
4    explicit MMElement(GstBaseTransform *gobj);
5
6    static void class_init(
7        Gst::ElementClass<MMElement> *klass);
8
9    Gst::FlowReturn transform_ip_vfunc(
10       const Glib::RefPtr<Gst::Buffer>& buf) override
11   {
12     std::cout << "Hello buffer!" << std::endl;
13
14     return FLOW_OK;
15   }
16 };
```

Base classes and interfaces

Classes:

- AudioBaseSink
- AudioFilter
- BaseSrc
- BaseTransform
- PushSrc
- VideoSink
- ...

Interfaces:

- Navigation
- URIHandler
- VideoOverlay
- ...

Plugins - properties

```
1  class MMElement : public Gst::BaseTransform
2  {
3     Glib::Property<bool> silent;
4     // ...
5  };
6
7  MMElement::MMElement(GstBaseTransform *gobj)
8  : Glib::ObjectBase(typeid(MMElement)),
9     Gst::Element(gobj),
10  silent( *this, "silent", false)
11 {
12 }
13
14 // ...
15  if (!silent)
16       std::cout << "Hello buffer!" << std::endl;
```

Properties - interface

```
Property (
  Glib::Object& object,
  const Glib::ustring& name,
  PropertyType& default_value,
  // since glibmm 2.49.1:
  const Glib::ustring& nick,
  const Glib::ustring& blurb,
  Glib::ParamFlags flags);
```

On property change signal:

```
  silent.get_proxy().signal_changed().connect([]{
    std::cout << "property changed" << std::endl;
  });
```

Other features

- convenient API of other classes (e.g. Caps, Bin, Iterator etc.)
- integration with gtkmm
- error handling: exceptions (instead of GError*)
- no need to copy boilerplatte
- less code
  - gst-template: C: 200 LOC, C++: 60 LOC
  - capsfilter: C: 550 LOC, C++: 350 LOC

Future

- gobject-introspection-based mm generator
- support for tracer plugins
- wrap missing classes/methods
- Microsoft Windows installer

Current release

- Version: 1.8.0
- Platforms:
  - Linux
  - Microsoft Windows:
    - MinGW
    - Microsoft Visual Studio 2015
- Dependencies:
  - glibmm-2.4 >= 2.47.6
  - libsigc++-2.0 >= 2.6.0
  - gstreamer-1.0 >= 1.8.0

Links

- Bugzilla:
  https://bugzilla.gnome.org/enter_bug.cgi?product=gstreamermm
- Mailing list:
  gtkmm-list@gnome.org
- GIT:
  https://git.gnome.org/gstreamermm/
- Releases:
  https://download.gnome.org/sources/gstreamermm/
- Documentation:
  https://developer.gnome.org/gstreamermm/