gstreamermm C++ wrapper

Marcin Kolny

marcin.kolny@gmail.com

October 8, 2015

Simple example

```
#include <gstreamermm.h>
#include <glibmm.h>
#include <iostream>

int main(int argc, char **argv)
{
  Gst::init(argc, argv);

  Glib::RefPtr<Gst::Pipeline> pipeline = Gst::Pipeline::create();
  Glib::RefPtr<Gst::Element>
    source = Gst::ElementFactory::create_element("videotestsrc"),
    sink = Gst::ElementFactory::create_element("xvimagesink", "video-sink");

  if (!source || !sink) {
    std::cerr << "One of the elements can not be create" << std::endl;
    return 1;
  }

  try {
    pipeline->add(source)->add(sink);
    source->link(sink);
  } catch (const std::runtime_error &err) {
    std::cerr << err.what() << std::endl;
    return 1;
  }

  pipeline->set_state(Gst::STATE_PLAYING);
  // ...
  pipeline->set_state(Gst::STATE_NULL);
  return 0;
}
```

Watching the bus

```
pipeline->get_bus()->add_watch(sigc::ptr_fun(&on_bus_message));

// ...

bool on_bus_message(const Glib::RefPtr<Gst::Bus>& bus,
    const Glib::RefPtr<Gst::Message>& message)
{
  switch(message->get_message_type())
  {
  case Gst::MESSAGE_EOS:
    mainloop->quit();
    return false;

  case Gst::MESSAGE_ERROR:
    {
      Glib::Error err = Glib::RefPtr<Gst::MessageError>::cast_static(message)->parse();
      std::cerr << "Error: " << err.what() << std::endl;
      mainloop->quit();
      return false;
    }

  default:
    return true;
  }
}
```

Properties

```
source->property("is-live", true)
      ->property("horizontal-spee", 10);

source->connect_property_changed("pattern", [source] {
  bool is_live;
  source->get_property("pattern", is_live);
  std::cout << "'pattern' property has been changed. "
     << "New value: " << is_live << std::endl;
});
```

Structure

```
Gst::Structure structure("struct-name",
    "field1", 42,
    "field2", CustomType("test"));
structure.set_field("field3", 11.6);

int field1;
CustomType field2;
double field3;
structure.get_field("field1", field1);
structure.get_field("field2", field2);
structure.get_field("field3", field3);

std::cout << field1 << " " << field2.get_data() <<
    " " << field3 << std::endl; // 42 test 11.6
```

Caps

```
Glib::RefPtr<Gst::Caps> caps =
    Gst::Caps::create_simple("video/x-raw",
                "width", 1024,
                "height", 768,
                "framerate", Gst::Fraction(25,1));

// ...

filter->set_property("caps", caps);
```

Signals

```
Player::Player()
{
  // ...
  decoder->signal_pad_added().connect(
      sigc::mem_fun(*this, &Player::decoder_pad_added));
}

void Player::decoder_pad_added(const RefPtr<Pad>& pad)
{
  RefPtr<Caps> caps = pad->get_current_caps();
  std::string media_type =
      caps->get_structure(0).get_name();

  if (std::regex_search(media_type, std::regex("^video")))
    pad->link(video_sink->get_static_pad("sink"));
  else
    std::cerr << "Unsupported media type" << std::endl;
}
```

Creating own elements

```
class SampleElement : public Gst::Element
{
  Glib::RefPtr<Gst::Pad> sinkpad;
  Glib::RefPtr<Gst::Pad> srcpad;
public:
  Glib::Property<int> buffer_cnt;
  explicit SampleElement(GstElement *gobj);
  static void base_init(
    Gst::ElementClass<SampleElement> *klass);

  Gst::FlowReturn chain(const Glib::RefPtr<Gst::Pad> &pad,

  void set_context_vfunc(
    const Glib::RefPtr<Gst::Context>& context) override;
  Gst::StateChangeReturn change_state_vfunc(
    Gst::StateChange transition) override;
  // ...
};
```

Creating own elements

```
static void SampleElement::base_init(Gst::ElementClass<SampleElement> *klass)
{
  klass->set_metadata("Sample Element",
        "Filter", "Count buffers", "Marcin Kolny <marcin.kolny@gmail.com>");
  klass->add_pad_template(Gst::PadTemplate::create("template", Gst::PAD_SINK,
      Gst::PAD_ALWAYS, Gst::Caps::create_any()));
}

Gst::FlowReturn SampleElement::chain(const Glib::RefPtr<Gst::Pad> &pad,
                                     Glib::RefPtr<Gst::Buffer> &buf)
{
  buffer_cnt = buffer_cnt + 1;
  return srcpad->push(std::move(buf));
}

SampleElement::SampleElement(GstElement *gobj)
: Glib::ObjectBase(typeid (SampleElement)),
  Gst::Element(gobj),
  buffer_cnt(*this, "buffer-cnt", 0)
{
  add_pad(sinkpad = Gst::Pad::create(get_pad_template("template"), "sink"));
  add_pad(srcpad = Gst::Pad::create(get_pad_template("template"), "src"));
  sinkpad->set_chain_function(sigc::mem_fun(*this, &SampleElement::chain));
}

static gboolean plugin_init (GstPlugin * plugin)
{
  Gst::init();
  return Gst::ElementFactory::register_element(Glib::wrap(plugin,true),
       "sampleelement", 10, Gst::register_mm_type<SampleElement>("SampleElement"));
}
```

Using base classes - GstBaseTransform

```
class SampleFilter : public Gst::BaseTransform
{
  static void base_init(
      Gst::ElementClass<SampleFilter> *klass);

  explicit SampleFilter(GstBaseTransform *gobj);
  virtual ~SampleFilter();

  Gst::FlowReturn transform_vfunc(
      const Glib::RefPtr<Gst::Buffer>& inbuf,
      const Glib::RefPtr<Gst::Buffer>& outbuf) override;
};
```

Using base classes - GstGLFilter

```
class SampleGLFilter : public Gst::Bad::GLFilter
{
public:
  static void base_init(
      Gst::ElementClass<SampleGLFilter> *klass);

  explicit SampleGLFilter(GstGLFilter *gobj);
  virtual ~SampleGLFilter();

  bool filter_texture_vfunc(
      guint in_tex,
      guint out_tex) override;
};
```

Current release

1.4.3

but 1.6 is coming!

Links

- Bugzilla:
  https://bugzilla.gnome.org/enter_bug.cgi?product=gstreamermm
- GIT:
  https://git.gnome.org/gstreamermm/
- Releases:
  http://ftp.gnome.org/pub/GNOME/sources/gstreamermm/
- Documentation:
  https://developer.gnome.org/gstreamermm/1.0/