A new era for GStreamer C++ bindings

GStreamer Conference 2025

23 October 2025

Sebastian Dröge <sebastian@centricular.com>

Centricular

gstreamer

This time we're not going to talk about Rust

Why C++?

- Existing codebases out there
  - Nobody is going to port all this legacy code to Rust
- Interop with C++ SDKs
  - NVIDIA, Qt, OpenCV, TensorFlow, Windows APIs, ...
- Arguably less bad than plain C

Why GStreamer C++ Bindings?

- C API **is** usable from C++ but
- Feels "foreign"
- No RAII memory management
- Less safe than necessary

Previous attempts

- gstreamermm
  - Unmaintained (6+ years)
  - Based on gtkmm (barely maintained)
  - Some questionable design decisions
  - Originally designed around C++98, now modernized
  - https://gitlab.gnome.org/GNOME/gstreamermm

Previous attempts

- qt-gstreamer
  - Unmaintained (10+ years)
  - https://cgit.freedesktop.org/gstreamer/qt-gstreamer/

Previous attempts

- cppgir
  - Didn't gain any traction
  - https://gitlab.com/mnauw/cppgir/
- Custom mini bindings
  - WebKit, ...

Hello peel!

- Started by Sergey Bugaev in 2024
- Gained quite some interest in GNOME
- https://gitlab.gnome.org/bugaevc/peel

peel overview

- Python code generator based on GObject Introspection
- Header-only, zero-dependency & zero-cost C++ API
  - No STL / RTTI / exception usage
  - No wrapper objects
- "Modern" C++ design practices
  - C++11 and above
  - Smart pointers
  - Don't emulate Rust, badly

peel overview

- Full availability of GObject APIs
  - Using C++ language features
  - Easy fallback to C APIs if needed
- Easy integration into meson and CMake build systems

```
Gst::init (&argc, &argv);

UniquePtr<GLib::MainLoop> loop = GLib::MainLoop::create (nullptr, false);

RefPtr<Gst::Element> pipeline =
  Gst::parse_launch ("audiotestsrc num-buffers=100 ! autoaudiosink", nullptr);

// Bus handling next slide

pipeline->set_state (Gst::State::PLAYING);

loop->run ();

pipeline->set_state (Gst::State::NULL_);
```

```
RefPtr<Gst::Bus> bus = pipeline->get_bus ();
bus->add_watch_full (G_PRIORITY_DEFAULT,
  [loop = static_cast<GLib::MainLoop *> (loop)]
  (Gst::Bus *bus, Gst::Message *msg) {
    switch (msg->type) {
      case Gst::Message::Type::ERROR_: {
        UniquePtr<GLib::Error> err;
        String debug_info;
        msg->parse_error (&err, &debug_info);
        GLib::print ("Error: %s\n", err->message);
        GLib::printerr ("Debugging information: %s\n",
            debug_info ? debug_info.c_str () : "none");
        loop->quit ();
        break;
      }
      case Gst::Message::Type::EOS:
        g_printerr ("EOS\n");
        loop->quit ();
        break;
      default:
        break;
      }

    return G_SOURCE_CONTINUE;
});
```

Smart Pointers

- Extensive use of smart pointers for memory management
- `RefPtr<T>` / `UniquePtr<T>`
  - Similar to `shared_ptr` / `unique_ptr`
  - Implement normal copy / move semantics
- `FloatPtr<T>` for *maybe* floating references
  - Avoids unnecessary refcounting
- `WeakPtr<T>` e.g. to break reference cycles
- `T *` for `transfer none` contexts

Classes, methods, ...

- All GObject types exposed as normal C++ classes / structs with methods
  - Same memory layout as C types
- No C++ RTTI
  - `element->cast<Gst::Bin> ()` instead of `dynamic_cast`
  - Upcasting automatic
- No C++ constructors
  - `create ()` static methods

Properties

- Static properties exposed via `Property` accessors
  - `obj->get_property (Gst::Object::prop_name ())`
  - Avoids type and name mismatches
- Dynamic properties via type selectors
  - `obj->get_property<peel::String> ("name")`
  - Also support for dynamic flags/enums

Signals & Callbacks

- Statically-typed connector/emitter functions
  - Support for signal details, before/after, ...
  - `element.connect_pad_added (...)`
- Dynamic signals can use C++ types too
  - Types need to match signal signature
  - `element.connect_signal ("pad-added", ...)`
- Signal handlers and all callbacks can use C++ lambdas
  - Be careful what exactly is captured!

Other types

- `String` / `const char*` for owned / unowned strings
- `UniquePtr<T[]>` / `ArrayRef<T>` / `T &[N]` for arrays
  - With known lengths but **no** bounds checks
- `ZTUniquePtr<T[]>` / `ZTArrayRef<T>` for zero-terminated arrays
  - also: `Strv` / `StrvRef` for zero-terminated string arrays
  - Not merged yet
- `GLib::List<T>` not implemented yet
  - Currently only `void *` items

C Interop

```
peel_nonnull_args (2)
peel::RefPtr<Element>
get_by_name (const char *name) noexcept
{
  ::GstBin *_peel_this = reinterpret_cast<::GstBin *> (this);
  ::GstElement * _peel_return = gst_bin_get_by_name (_peel_this, name);
  return peel::RefPtr<Element>::adopt_ref (reinterpret_cast<Element *> (_peel_return));
}
```

Defining new GObjects

- Possible by defining a new C++ class and using some macros
- Overriding virtual methods not like in C++
- Non-static destructor maps to `GObject::finalize`
- Transparent from C POV
  - Instantiation / usage from any language possible

Defining new GObjects

```
class MyClass final : public Object
{
  PEEL_SIMPLE_CLASS (MyClass, Object)

public:
  static RefPtr<MyClass> create (void)
  {
    return peel::GObject::Object::create<MyClass> ();
  }
};

PEEL_CLASS_IMPL (MyClass, "MyClass", Object)

void
MyClass::Class::init ()
{
}
```

Overriding GObject vfuncs

```
void
MyClass::vfunc_dispose ()
{
  parent_vfunc_dispose<MyClass> ();
}

void
MyClass::Class::init ()
{
  override_vfunc_dispose<MyClass> ();
}
```

Status

- GStreamer core/base/audio/video/pbutils/app working
  - Others likely work too
  - Not widely tested yet
- A couple of unmerged MRs required
- Various features still missing
  - Better miniobject APIs
  - Typed `GLib::List`
  - …
- Generally ready for usage

Personal conclusion

- peel-style C++ is a lot less horrible than GObject-style C
  - Yes, even (especially!) when using `g_autoptr`
- I still prefer Rust wherever I can use it
- Take ideas learned here back to the Rust bindings / codegen

Questions? Comments?

Code available at

- https://gitlab.gnome.org/bugaevc/peel
- https://gitlab.gnome.org/sdroege/peel/-/commits/gst-wip
- https://gitlab.freedesktop.org/slomo/gstreamer-peel-examples