Multimedia using Rust & GStreamer

Multimedia using Rust and GStreamer

Sanchayan Maity

Who?

- Consultant Software Engineer @ asymptotic
  - Open source consulting firm based out of Toronto, Bangalore & Hyderabad
  - Work on low level systems software centred around multimedia
  - GStreamer, PipeWire, PulseAudio
- Embedded Systems background
- C, Rust and Haskell
- Organizing Rust and Haskell meetup Bangalore since 2018

Agenda

- Introduction to GStreamer
- Why Rust
- Rust and GStreamer

GStreamer

- Multiplatform Pipeline based multimedia framework
- Bindings for various languages
- Supported on Linux, macOS, Android and Windows
- Allows building complex media processing workflows
- Some applications
  - GstLAL (gravitational wave data analysis)
  - PiTiVi (Video Editor)
  - amaroK, Banshee, Clementine (audio players)
  - Empathy (VOIP and video conferencing)
  - Rygel (DLNA streaming server and renderer)
  - Showtime, Clapper, Totem (Media players for desktop)

Simple pipeline

```
gst-launch-1.0 videotestsrc ! autovideosink
gst-launch-1.0 audiotestsrc ! autoaudiosink
```

gst-inspect

```
Factory Details:
  Rank                     none (0)
  Long-name                Video test source
  Klass                    Source/Video
  Description              Creates a test video stream
  Author                   David A. Schleef <ds@schleef.org>
  Documentation            https://gstreamer.freedesktop.org/documentation/videotestsrc/#
videotestsrc-page

Plugin Details:
  Name                     videotestsrc
  Description              Creates a test video stream
  Filename                 /usr/lib/gstreamer-1.0/libgstvideotestsrc.so
  Version                  1.24.9
  License                  LGPL
  Source module            gst-plugins-base
  Documentation            https://gstreamer.freedesktop.org/documentation/videotestsrc/
```

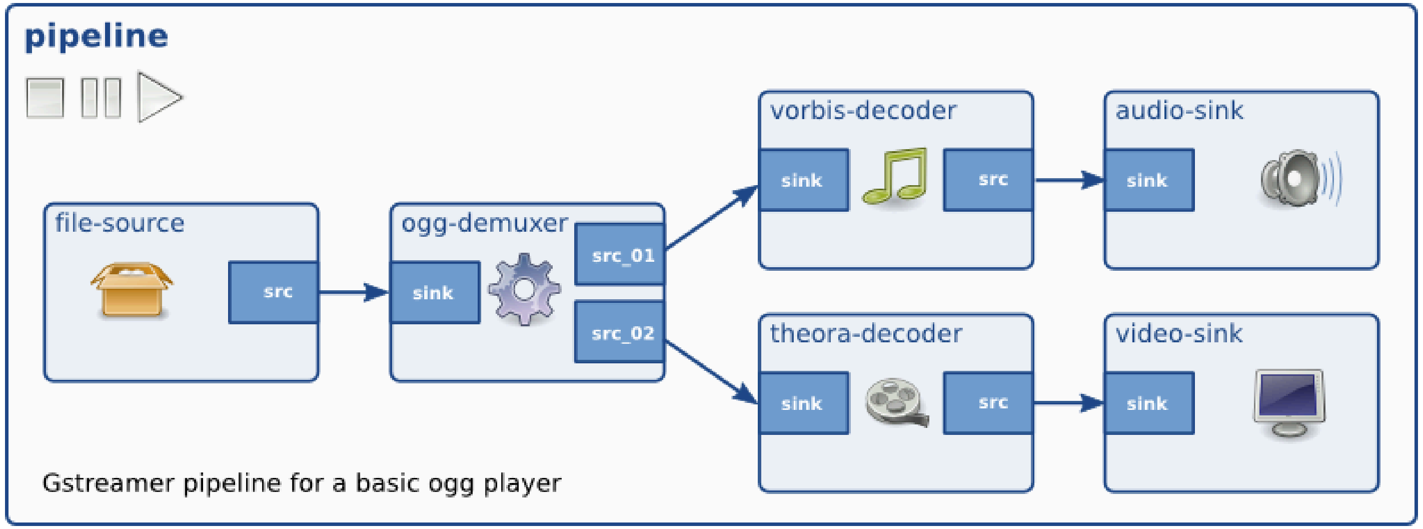
gst-inspect (ii)

```
  Source release date      2024-10-30
  Binary package           Arch Linux GStreamer 1.24.9-3
  Origin URL               https://www.archlinux.org/

GObject
 +----GInitiallyUnowned
       +----GstObject
             +----GstElement
                   +----GstBaseSrc
                         +----GstPushSrc
                               +----GstVideoTestSrc

Pad Templates:
  SRC template: 'src'
    Availability: Always
    Capabilities:
      video/x-raw
```

Media pipeline



Why Rust?

- Codec implementations in pure Rust (Rust Audio, Xiph AV1, Symphonia)
- Things to care about
  - **Low cognitive overhead**
  - Immutability
  - Expressive type system
  - Memory safety and concurrency
  - Foreign Function Interface

Why Rust?

- Bindings/abstractions over GLib/GObject and for GStreamer[1]
- Provides a root for the object hierarchy tree filed in by the GStreamer library
- Gives basic reference counting, parenting functionality and locking.
- GObject
  - GstObject
    - GstAllocator
    - GstBufferPool
    - GstBus
    - GstClock
    - GstDevice
    - GstDeviceMonitor
    - GstDeviceProvider
    - GstElement
    - GstPad

[1]GstObject

Why immutability and types matter?

```
let caps: gst::Caps = gst::Caps::builder("video/x-raw")
    .field("width", crop_w)
    .field("height", crop_h)
    .field("pixel-aspect-ratio", gst::Fraction::new(1, 1))
    .build();
let s = caps.remove_structure(0);
```

Why immutability and types matter?

```
warning: unused variable: `s`
   --> video-bin/src/imp.rs:152:13
    |
152 |         let s = caps.remove_structure(0);
    |             ^ help: if this is intentional, prefix it with an
                      underscore: `_s`
    |
    = note: `#[warn(unused_variables)]` on by default
error[E0596]: cannot borrow data in dereference of `gstreamer::Caps`
              as mutable
   --> video-bin/src/imp.rs:152:17
    |
152 |         let s = caps.remove_structure(0);
    |                 ^^^^^^^^^^^^^^^^^^^^^^^^ cannot borrow as mutable
    |
    = help: trait `DerefMut` is required to modify through a dereference,
            but it is not implemented for `gstreamer::Caps`
```

Why immutability and types matter?

```
let mut caps: gst::Caps = gst::Caps::builder("video/x-raw")
    .field("width", crop_w)
    .field("height", crop_h)
    .field("pixel-aspect-ratio", gst::Fraction::new(1, 1))
    .build();
let _s = caps.remove_structure(0);
```

Why immutability and types matter?

```
warning: variable does not need to be mutable
   --> video-bin/src/imp.rs:147:13
    |
147 |         let mut caps: gst::Caps = gst::Caps::builder("video/x-raw")
    |             ----^^^^
    |             |
    |             help: remove this `mut`
    |
    = note: `#[warn(unused_mut)]` on by default
error[E0596]: cannot borrow data in dereference of `gstreamer::Caps`
              as mutable
   --> video-bin/src/imp.rs:152:18
    |
152 |         let _s = caps.remove_structure(0);
    |                  ^^^^^^^^^^^^^^^^^^^^^^^^ cannot borrow as mutable
    = help: trait `DerefMut` is required to modify through a dereference,
            but it is not implemented for `gstreamer::Caps`
```

Why immutability and types matter?

```rust
let caps: gst::Caps = gst::Caps::builder("video/x-raw")
    .field("width", crop_w)
    .field("height", crop_h)
    .field("pixel-aspect-ratio", gst::Fraction::new(1, 1))
    .build();
let caps = caps.get_mut().unwrap();
let _s = caps.remove_structure(0);
```

Why immutability and types matter?

```
error[E0596]: cannot borrow `caps` as mutable, as it is not declared
              as mutable
   --> video-bin/src/imp.rs:152:20
    |
147 |         let caps: gst::Caps = gst::Caps::builder("video/x-raw")
    |             ---- help: consider changing this to be mutable:
                         `mut caps`
...
152 |         let caps = caps.get_mut().unwrap();
    |                    ^^^^^^^^^^^^^^ cannot borrow as mutable

For more information about this error, try `rustc --explain E0596`.
```

Why immutability and types matter?

```
let mut caps: gst::Caps = gst::Caps::builder("video/x-raw")
    .field("width", crop_w)
    .field("height", crop_h)
    .field("pixel-aspect-ratio", gst::Fraction::new(1, 1))
    .build();
if let Some(caps) = caps.get_mut() {
    let _s = caps.remove_structure(0);
}
```

Code

```
let src = gst::ElementFactory::make("filesrc")
    .property("location", "sample.ogv")
    .build()
    .unwrap();
let demux = gst::ElementFactory::make("oggdemux").build().unwrap();

let pipeline_weak = pipeline.downgrade();
demux.connect("pad-added", false, move |args| {
    let pipeline = match pipeline_weak.upgrade() {
        Some(self_) => self_,
        None => return None,
    };

    let pad = args[1]
        .get::<gst::Pad>()
        .expect("Second argument to demux pad-added must be pad");
```

Code

```
if let Some(caps) = pad.current_caps() {
    let s = caps.structure(0).unwrap();

    let (decoder, sink) = if s.name().starts_with("video") {
        let decoder = gst::ElementFactory::make("theoradec").build().unwrap();
        let sink = gst::ElementFactory::make("autovideosink").build().unwrap();
        (decoder, sink)
    } else {
        let decoder = gst::ElementFactory::make("vorbisdec").build().unwrap();
        let sink = gst::ElementFactory::make("autoaudiosink").build().unwrap();
        (decoder, sink)
    };

    let queue1 = gst::ElementFactory::make("queue").build().unwrap();
    let queue2 = gst::ElementFactory::make("queue").build().unwrap();
```

Code

```
        pipeline
            .add_many([&queue1, &decoder, &queue2, &sink])
            .unwrap();

        let sinkpad = queue1.static_pad("sink").unwrap();
        pad.link(&sinkpad).unwrap();

        queue1.link(&decoder).unwrap();
        decoder.link(&queue2).unwrap();
        queue2.link(&sink).unwrap();

        queue1.sync_state_with_parent().unwrap();
        decoder.sync_state_with_parent().unwrap();
        queue2.sync_state_with_parent().unwrap();
        sink.sync_state_with_parent().unwrap();
    }
```

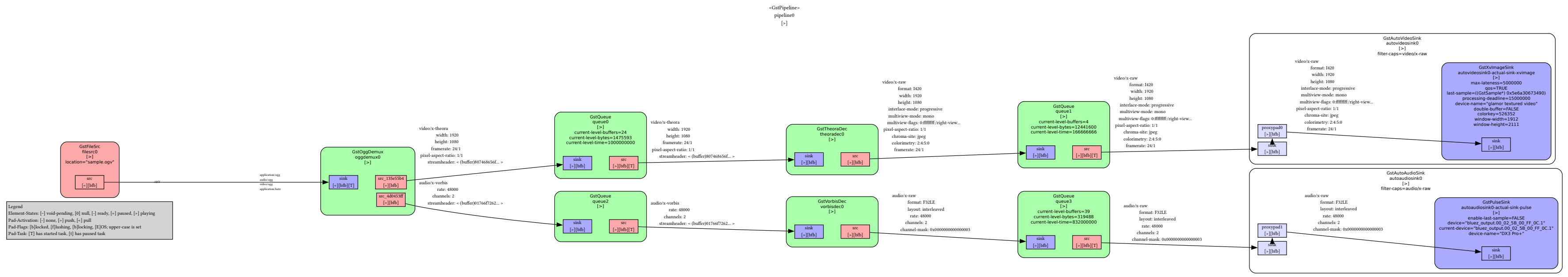
Code

```
        None
    });

    pipeline.add_many([&src, &demux]).unwrap();

    src.link(&demux).unwrap();
```

Media pipeline



Some stats

- **gstreamer-rs** & **gst-plugins-rs**[2]
  - gstreamer-rs: ~2700 commits, gst-plugins-rs: ~2600 commits
  - gstreamer-rs: ~85 contributors, gst-plugins-rs: ~110 contributors
  - gst-plugins-rs: ~ +180k SLOC / −37k SLOC
  - gst-plugins-rs: Overall 47 plugins, 149 elements
- In relation to the GStreamer monorepo
  - 1.22 cycle: ~33% commits / MRs in Rust modules
  - 1.24 cycle: ~25% commits / MRs in Rust modules

[2] GStreamer & Rust: What has happened over the last 5 years

Resources

- Dynamic Pipelines
- GObject subclassing in Rust
- GStreamer bindings for Rust
- Rust GStreamer Plugins
- Using GStreamer
- How to get started with GStreamer
- GStreamer for your backend services
- OGG demultiplexing Rust sample code

Questions?

- Rust Bangalore
  - Meetup: https://hasgeek.com/rustbangalore
  - Telegram: https://t.me/RustIndia
- Reach out on
  - email:

    ```
    - me@sanchayanmaity.net
    - sanchayan@asymptotic.io
    - hello@asymptotic.io
    ```

  - Mastodon: sanchayanmaity.com
  - Blog: sanchayanmaity.net