

Color, tracing and WebAssembly

A year of GStreamer for video editing

Thibault Saunier

GStreamer conference 2026 - Prague

Plan

- Color grading support in GES
- Made GES APIs MT safe!
- Performance
- GES in the browser in wasm

Color correction: float video formats

- Post-production APIs/formats work on floating point pixels
formats: OpenEXR, OpenColorIO, ACES
- GStreamer had no float pixel format
 - We added `RGBA_F16` and `RGBA_F32` in GstVideo
- Others built on it afterward: float rework, HIP, onnx

OpenColorIO

- Open source color management library, used by many post-production tools
- A config describes the color spaces and the transforms between them (matrices, curves, LUTs)
- In GStreamer: `glociobin` and one GL element per grading node
- All the nodes composed into a single GL pass
- Reorderable, keyframeable, saved in xges
- Prototype, not upstream yet

HDR from the editor to the browser

- Preview and render convert the same way
- HDR carried through the WebRTC preview to the browser
- Delivered as HEVC: Chrome has no HDR AV1 over WebRTC

The GES API is thread safe

- Internal locking in every public function
- A `timeline_lock` protects the timeline of each element
- `_full` variants of the getters that returned borrowed references
- The `GES_TIMELINE_ELEMENT_*` getter macros go through the locked getters

Optimizing GES: where is time spent

- Timed traces: how long operations take
- Emitted by the code itself
- Tracers in `gst-plugins-rs` turn spans into Rust `tracing` spans
- A subscriber writes them out, e.g. a Perfetto trace
- Nothing is evaluated when no tracer listens

Using spans in C

```
GST_DEFINE_TRACE_FORMAT (gstelement_set_state,  
    "element-name", STRING, "state", STRING)  
  
GstStateChangeReturn  
gst_element_set_state (GstElement * element, GstState sta  
{  
    GstStateChangeReturn result;  
    GST_TRACE_SCOPE_BEGIN (gstelement_set_state,  
        STRING (GST_ELEMENT_NAME (element)),  
        STRING (gst_state_get_name (state)));  
    result = do_the_work (element, state);  
    GST_TRACE_SCOPE_END (gstelement_set_state);  
    return result;  
}
```

One memory type end to end

- `GESVideoElementSelector` picks the compositor and converters from the registry
 - new `Uploader`, `Downloader`, `Converter`, `Scaler` classifications
- One memory type from decoder to timeline src pad

Demo: grading, pipeline and trace

GES in the browser

- A first version works: a GES timeline plays in the page
- Uses emscripten: a C compiler toolchain with an emulation of unix APIs
- WebCodecs decode, WebGL compositing

Built on gst.wasm

- Fluendo's gst.wasm (fluendo.github.io/gst.wasm): the groundwork and the first browser elements
- I moved some of those elements to the GStreamer monorepo
- A Meson cross file for the build

What was needed to get GES working properly

- Web platform limitations to work around:
 - Strict function signatures
 - Canvas and WebCodecs calls on one thread, the OffscreenCanvas one
 - `GstBus` and `GstPoll` wake-ups reworked

What do I have

Demo: GES GL compositing validate test, in the browser

- Timelines play in the web browser
- Tests run in the page, through a GstValidate runner
- On branches: GStreamer, GLib

Questions?