GStreamer State of the Union

GStreamer Conference, Dublin
8 October 2015

Tim-Philipp Müller <tim@centricular.com>

Centricular

Introduction

- GStreamer core developer, co-maintainer, backseat release manager
- have been using and hacking on GStreamer for more than 10 years, both for fun and profit
- Centricular co-founder

What is GStreamer ?

- you know this, so key points only

What is GStreamer ?

- framework for multimedia processing
- cross-platform, toolkit agnostic
- any and all use cases
- set of libraries and plugins
- abstract API, very extensible
- often wrapping other libraries (for decoders, encoders, filters, etc.)

What is GStreamer ? (cont'd)

- low-level API and high-level API
  - playbin, encodebin, RTSP server, non-linear editing, VoIP etc.
- integration with other frameworks and projects
  - e.g. WebKit/Blink, Clutter, Cogl, OpenGL, Windows, OS X, Android, iOS
  - goal is to adapt to and integrate with other platforms and frameworks (inputs, outputs, decoders, DSPs/GPUs..)

So, what has everyone been up to?

We finally got a release out!

Loads of new features

- Stereoscopic 3D and multiview video support
- Trick mode API
- Improved DTS/PTS handling for negative DTS
- New video converter API: faster + more correct conversion + scaling of raw video formats

New features - II

- splitmux: chunked recording of MP4, Matroska, MPEG-TS
- MP4/MOV muxer: new "robust" mode
- Content Protection API and Common Encryption (CENC) support for DASH/MP4
- Loads of other adaptive streaming (DASH, HLS and MSS) improvements

New features - III

- RTSP server RECORD support
- Retransmissions (RTX) support in RTSP server + client
- RTSP seeking works properly now
- RTCP scheduling improvements + reduced size RTCP
- SRTP/DTLS support (WebRTC)

New features - IV

- v4l2src renegotiation + v4l2transform scaling
- Live mixing support in audiomixer + compositor that works
- Blackmagic Design DeckLink capture and playback card support rewritten from scratch, incl. 2k/4k support and mode sensing
- PTP + NTP network clocks and better remote clock tracking
- KLV metadata support in RTP and MPEG-TS

New features - V

- OpenGL improvements:
  - OpenGL3
  - 3D video
  - multiple contexts and context propagation
  - transfer/conversion separation (performance)
  - subtitle blending

- New OpenGL video sinks: QML, GtkGL, CoreAnimation

- High-quality text subtitle overlay at display resolutions with glimagesink or gtkglsink

New features - VI

- GstHarness API for unit tests
- gst-validate: new plugin system, concurrent test runs, valgrind support
- cerbero build tool: new 'bundle-source' command
- Android, iOS, OS X and Windows improvements

Release notes

More info:

http://gstreamer.freedesktop.org/releases/1.6/

Recently landed in git (post-release)

- new GstTracer tracing framework merged
- DASH: official DASH-IF test vectors all work now! (except those that are broken, of which there are a few, but at least one can file bugs)
- Nvidia NVENC hardware-accelerated encoding support
- QML video sink works on Android + iOS now
- GstContext propagation via bin hierarchy

Coming up in 1.8

- GL shader changes: more flexible
- Android camera source
- Android zero-copy decoding
- dmabuf support
- video4linux encoder

Coming up in 1.8 (cont'd)

- RTSP: server sink + client (RECORD) sink
- RFC7273: media clock info in SDP for automatic clock synchronisation between multiple devices without configuration (AES-67/Ravenna broadcasting standard)
- live input handling for muxers (maybe)

Coming up in 1.8 (cont'd)

- easier sandboxing (maybe)
- better subtitle support
- gst-player
- and much more

Enough with features, what else is new?

Enough with features, what else is new?

- Quality Assurance (QA)
  - build bots very helpful, esp. for non-Linux platforms
  - gst-validate test suite slowly beginning to be useful
    - testing still quite basic
    - need more tests: different transports/protocols/features (help!)

But not everything is perfect.

Admission: I tried to crowdsource this talk

And asked people what they thought GStreamer's biggest problems or challenges were.

(I wasn't brave enough to ask on twitter though.)

Here's what they said

- "*You don't release often enough!*"
- "*There are too many people who are not yet using GStreamer*"
- "*GStreamer sucks! (But it doesn't have to)*"

"*You don't release often enough!*"

Let's change that!

Let's do fixed date 6-monthly release cycles!

Release Schedule for 1.8

18 Dec '15: 1st pre-release

8 Jan '16: 2nd pre-release + freeze for major changes

22 Jan '16: rc1 + full freeze

29 Jan '16: rc2 (if needed)

5 Feb '16: 1.8.0 release

Public Service Announcement

If you don't get your stuff in on time,
it will just have to go in next cycle.

If stuff that's in turns out not to be quite ready, we should just back it out if we can't fix it up quickly.

Don't wait until the third pre-release to start thinking about everything you wanted to merge that cycle.

"*There are too many people who are not yet using GStreamer*"

Why is that?

Because we don't make it easy for them!

What can we do better?

People don't like to write code with GLib/GObject.

--> more higher-level APIs that don't leak GObject

e.g. gst-player

--> more convenient API for common tasks

e.g. splitmuxsink, camerabin etc.

What can we do better?

Windows:

- .net => gstreamer-sharp
- Visual Studio integration

What can we do better?

Android:

- no canonical Java bindings
- technical: zerocopy decoding; camera source (both in progress, rsn)
- how to use GStreamer stuff in Android Studio
- we tend to use command line tools, but that doesn't match what other Android devs use; we don't really know how to improve this exactly - help!

What can we do better?

OS X + iOS:

- the way we package GStreamer is not quite right, but we don't know what to fix exactly - help!

What can we do better?

Linux:

- hardware-accelerated decoding story on the desktop still far from perfect, causes lots of problems for people instead of just working out of the box

- better integration with lower levels (dmabuf, etc.)

- but: QML + GTK GL sinks are great!

What can we do better?

Our development story is broken:

- gst-uninstalled only works when developing on the target machine (and Linux/OSX probably)
- Cerbero: Because Everything Else Is Worse
  - builds binaries, but painful for development
  - needs docs (old story)
- perhaps it's time for a different build system?

What can we do better?

"Killer app(s)" to showcase GStreamer's capabilities

Would be nice to at least package something based on gst-player for the various platforms.

Anyone feel like helping out?

What else can we do better?

- Developer Tools (old story)
  - hopefully soon now that GstTracer has been merged
  - also see lightning talk about remote debugging
- Daily builds (soon hopefully)

We still need better docs !

- more docs
- different docs (tutorials, howtos)
- demo code/apps written as examples
- GStreamer book

Other Challenges

- Internet of Things: smaller binary size and footprint for IoT devices
- Scalability at the other end (thousands of threads..)
- Anything else? What do you think? Let us know!

Conclusion

- GStreamer is technically better, more stable, more featureful, and more flexible than ever
- feature gaps are being plugged
- more work needed on improving developer experience, tools and documentation, especially on non-Linux systems
- you can help!

Thank you and enjoy the conference !

Questions or Comments ?