GStreamer 1.0

FOSDEM, Brussels
4 February 2012

Tim-Philipp Müller <tim.muller@collabora.co.uk>

<span weight="light">Tim-Philipp Müller &lt;tim.muller@collabora.co.uk&gt;</span>

Introduction

- who am I ?
- what is GStreamer ?
- 0.10, 0.11, 1.0 and all that
- structure of this talk

What is GStreamer ?

- set of libraries
- pipeline-based: elements, components, pads, data flow
- plugins, abstract API
- often wrap other libraries (decoders, encoders, filters, etc.)

What is GStreamer ? (cont'd)

- low-level API and high-level API
  - playbin2, encodebin, gst-editing-services, gst-rtsp-server
- integration with other frameworks and projects
  - e.g. webkit, android, clutter
  - goal is to adapt to platform/framework (inputs, outputs, decoders, etc.)

Where has GStreamer been used ?

- music players, video players
- web browsers (html5) • webkit-gtk/epiphany, opera
- cameras (e.g. Axis)
- transcoding (transmageddon, arista, Rygel DLNA server)
- streaming servers (flumotion, gst-stream-server, gst-rtsp-server)
- audio post-processing and editing (jokosher)
- video post-processing and editing (pitivi, novacut)
- communication: voice/video calls (farsight/telepathy, empathy, Tandberg Movi)
- video walls (Barco, Oblong)

Versions and all that

- API/ABI compatibility
- stable: 0.6.x, 0.8.x, 0.10.x, 1.0.x
- development: 0.9.x, 0.11.x
    - 0.9 --> 0.10
    - 0.11 --> 1.0
- lots of cool stuff has been happening in 0.10 lately, but I'm not going to talk about that

Structure of this talk

- Introduction ☑
- GStreamer 1.0
  - what we're trying to fix
  - major changes
- what does it all mean for ...
  - users?
  - application developers?
  - plugin developers?
  - bindings developers and users?
  - for distributors?

GStreamer 1.0

- been talking about it for a long time, since around 2007
- 0.10 just worked too well, kept extending it and extending it
- but at some point it just became too awkward
  - people accumulated hacks to work around limitations
  - people kept pushing the framework, doing lots of cool things that people weren't doing at that scale when 0.10 was in the works
  - we just couldn't do some stuff we wanted to do
  - many things could be done, but were just too complex to do

Challenges

- embedded or mobile platforms
  - quirks/customisation
  - efficiency even more critical
  - lower cpu power, specialised processing units (DSPs, GPUs)

- GPUs also on the rise as extra processing units on the desktop

  - video decoding/encoding/transcoding - intel, ati, nvidia
  - GPU image processing
  - more efficient, lower power consumption
  - can be done in 0.10, but it's not very nice

- dynamic pipelines

- renegotiation (format changes while pipeline is running)

Better Memory Management

- CPU memory/cache --- PCIe bus --- GPU graphics memory

- want to avoid copying memory back and forth from the CPU to the GPU to the CPU and back

- want better control over what memory is where and who's using what when

- currently we don't handle that very well

    - very little control or knowledge

    - just pass pointers + size arguments about, but don't actually know when plugins read or write to that; there are mechanisms to ensure a buffer is safe to write to, but if a plugin forgets to call the right function, we will never know and heisenbugs may or may not occur

Better Memory Management: video

- strides
- plane offsets
- padding
- alignment

- currently in 0.10 strides + plane offsets + padding are fixed per format (given a certain width/heigh), which doesn't always map to the hardware. When it doesn't, we have to memcpy, which is very inefficient.

- could be worked around in 0.10, but it's not very nice (custom media types that won't interact well with other existing elements, etc.)

- 1.0 is much more flexible about how the pixels are laid out in memory, strides, offset, padding and alignment can be negotiated between elements

- better memory management = better integration + performance

Better Memory Management: video (cont'd)

- delayed processing to avoid processing, or do multiple operations in one go, minimising memory access. This is still pretty much experimental.
  - e.g. cropping, just tag it and do it when rendering/encoding
- add extra information to video buffers, like region of interest (e.g. from face tracking)
- can tag fields/frames properly and more flexibly (3D video, interlaced content)
- but not just about efficiency, also correctness
  - e.g. proper colourspace information

Better Memory Management: other features

- memory allocators
- bufferpools
  - were used before, but the other elements ususally didn't know about that, now those can be negotiated and configured by the other involved elements

Renegotiation

- what's the problem?
  - in 0.10, if it worked at all or not depended a lot on the exact elements involved
- negotiation was linked with buffer allocation, which only works in one direction, which is a problem if the upstream element / source operates a buffer pool, for example (v4l2src) or no allocation is done (mp3parser)
- in 1.0, negotiation is completely decoupled from the buffer allocation

Dynamic pipelines

- could be done in 0.10, but was very very cumbersome and required good understanding of all the low-level plumbing and conventions in GStreamer. pad blocking; new segment events, etc.

- e.g. cheese camera / photobooth application: dynamically change/add/remove effect filters, add encoders/muxers/filesink to record video to a file.

- compressed format passthrough in pulseaudio (mp3 decoding => bluetooth headset plugged in, pulseaudio tells gstreamer's pulsesink, pulsesink tells upstream elements like decodebin, which then reconfigures itself to remove the mp3 decoder and pass the mp3 data straight through to pulse)

- solution: sticky events for replay, timing knobs on pads (offset)

Some other improvements

- explicit PTS/DTS timestamps on buffers
- Buffer metadata
- lots of API clean-ups (API, caps, base classes)
- simpler caps (raw video/audio caps, channel positions, stream headers etc.)
- GstController now much more tightly integrated with GstObject and bindings-friendly

Current 1.0 status

- core/base ported
- many other elements ported
- still some important elements missing (e.g. deinterlace)
- totem/clutter-gst tentatively ported
- transmageddon in process

- status reports Wim has sent to mailing list

- porting guide in docs/random/porting-to-0.11.txt

So … how much faster is it then?

- too early for numbers really
- few plugins currently make use of the new features
- generally at least as fast as 0.10, this is before making use of new features and fixing silly bugs
- important thing is that we know we can fix/improve things

What does it mean for users?

- not much really, new features in the future
- applications will hopefully mostly migrate in the course of the next 12 months or so

What does it mean for application developers?

- very little change (unless you use the python bindings)
- still have elements, pads, set states
- get messages on GstBus
- queries, events
- pad probes: new and improved
- caps: some have changed a bit (e.g. raw video/audio), most stay the same
- porting guide
- when to start porting?

What does it mean for bindings developers and users?

- depends
- gobject-introspection
- make things easier for bindings, in theory
- big change for python people, need to port from 0.10 to 0.11/1.0 API, and at the same time from pygst to py-gi

what does it mean for distributors?

- 0.10 and 0.11/1.0 can be installed in parallel
- some applications can use the old 0.10 and some can use 0.11
- but probably can't use both versions in parallel in the same process
- when should you start packaging 0.11?
- new versioning / bug-fix releases

Questions?

Thank You!

Pictures

TGV bending by aleske (http://www.fotopedia.com/items/flickr-2160201830)