GStreamer

Florian “Florob” Zeitz

2017-02-15

What is GStreamer?

- multimedia framework
- GObject-based
- modular/plugin architecture
- current versions are 1.x, 0.10.x is still being phased out

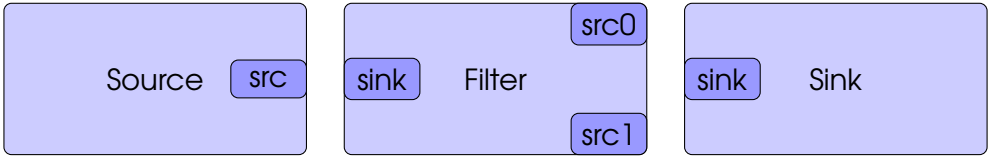
The Plugin Zoo — Welcome to the Jungle

- *gst-plugins-base*: basic set of well-supported plugins
- *gst-plugins-good*: well-supported plugins under "preferred" license
- *gst-plugins-ugly*: well-supported plugins under icky license
- *gst-plugins-bad*: less supported plugins
- *gst-libav*: wrappers around libav

Concepts

- data-flow is modelled as a *pipeline*
- pipelines are build from *elements*
- elements connect via their *pads*
- valid connections are determined by a pads *capabilities*
- an element collecting other elements is called a *bin*

Elements



- *Source*: Only has outputs
  e. g. filesrc, jackaudiosrc, autoaudiosrc, audiotestsrc, tcpclientsrc
- *Sink*: Only has inputs
  e. g. filesink, jackaudiosink, autoaudiosink, fakesink, tcpserversink
- *Filter*: Has inputs and outputs
  e. g. flacdec, opusenc, matroskamux, audioconvert, decodebin

Pads and Capabilities

- *static*: always available
  e. g. filesrc.src
- *dynamic*: sometimes available
  e. g. demux outputs
- *request*: available on request
  e. g. mux inputs, tee outputs

```
SRC template: 'src'
  Availability: Always
  Capabilities:
    audio/x-opus

SINK template: 'sink'
  Availability: Always
  Capabilities:
    audio/x-raw
                format: S16LE
                layout: interleaved
                  rate: 48000
              channels: [ 1, 8 ]
    audio/x-raw
                format: S16LE
                layout: interleaved
                  rate: { 8000, 12000, 16000, 24000 }
              channels: [ 1, 8 ]
```

Finding Elements

- `gst-inspect-1.0` lists all elements
- `gst-inspect-1.0 <plugin>` lists all elements in a plugin
- `gst-inspect-1.0 <element>` lists details about an element

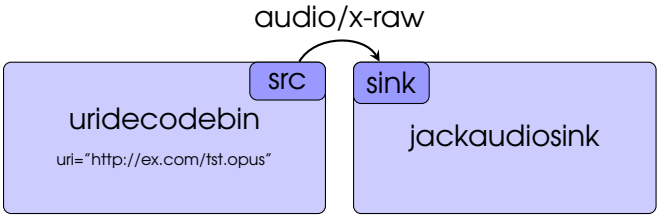
gst-launch

- simple pipelines can be launched with `gst-launch-1.0 <pipeline-desc>`
- elements are described by `elem-type [prop=value]*`
- named elements can be referenced by `elem-name.`
- pads can be referenced by `elem-name.pad-name`
- connections are defined using `!`

Play something, play anything


```
gst-launch-1.0 playbin uri="http://example.com/test.opus"
```

Play something, play anything, to JACK



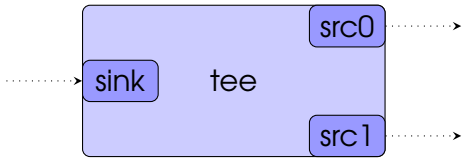

```
gst-launch-1.0 uridecodebin uri="http://ex.com/tst.opus" !
(audioconvert ! audioresample !)
jackaudiosink
```

Convert FLAC to Opus


```
gst-launch-1.0 filesrc location="in.flac" !
decodebin ! audioresample ! opusenc ! oggmux !
filesink location="out.opus"
```

- audioresample, because Opus doesn't support 44100
- transfers metadata

Display some visualization




```
gst-launch-1.0 uridecodebin uri="file://…" ! tee name=t
t. ! queue ! libvisual_jess !
videoconvert ! autovideosink
t. ! queue ! autoaudiosink
```

- tee multiplies the signal
- queue is a buffer, allowing parts of the pipeline to run independently

GObject Introspection

- introspection for GObject-based libraries
- used to generate bindings (compiled or on the fly)

```
1 import gi
2
3 gi.require_version('Gst', '1.0')
4 from gi.repository import Gst
```

Basic usage


```
1  import gi
2
3  gi.require_version('Gst', '1.0')
4  from gi.repository import Gst
5  from gi.repository import GObject
6
7  Gst.init(None)
8
9  pipe = Gst.parse_launch("playbin uri=http://ex.com/tst.opus")
10 pipe.set_state(Gst.State.PLAYING)
11
12 loop = GObject.MainLoop()
13 loop.run()
```

Buses

- Send messages around
- per pipeline and per element

```
1 pipe.bus.add_signal_watch()
2 pipe.bus.connect("message::element", elem_cb)
3 pipe.bus.connect("message::eos", eos_cb)
4 pipe.bus.connect("message::error", error_cb)
```

DEMO

Resources

- GStreamer documentation

Thank you for your attention. Any questions?

https://babelmonkeys.de/~florob/talks/OSAMC-2017-02-15-gstreamer.pdf