Gstreamer Editing Services

Video Editing in your pocket

(size of pocket not specified)

Edward Hervey
edward.hervey@collabora.co.uk

Edward Hervey

- Co-founder of Collabora Multimedia
- FLOSS user since 1995
- GStreamer Hacker since 2003
- PiTiVi video editor
- French (despite not striking)

Primary goal of GES project

- **Provide everything needed to make editing applications trivial to write**
- GNonLin/GStreamer were not enough
- Take GES + GStreamer
- Sprinkle UI on top of it
- => You have an editor !

Secondary Goals

- **Complete high-level solution**
  - Not just editing, but also playback, encoding, media discovery, ...
- **Flexible**
  - Not just one use-case in mind
- **NO HACKS !**
  - Upstream as much as possible

Breakdown

- GES high-level library
  - Timeline, Layer, Track
  - Convenience objects
- Peripheral libraries/improvements
  - libgstprofile, encodebin
  - GstDiscoverer
  - ...
- Lessons learnt
- Ideas and improvements

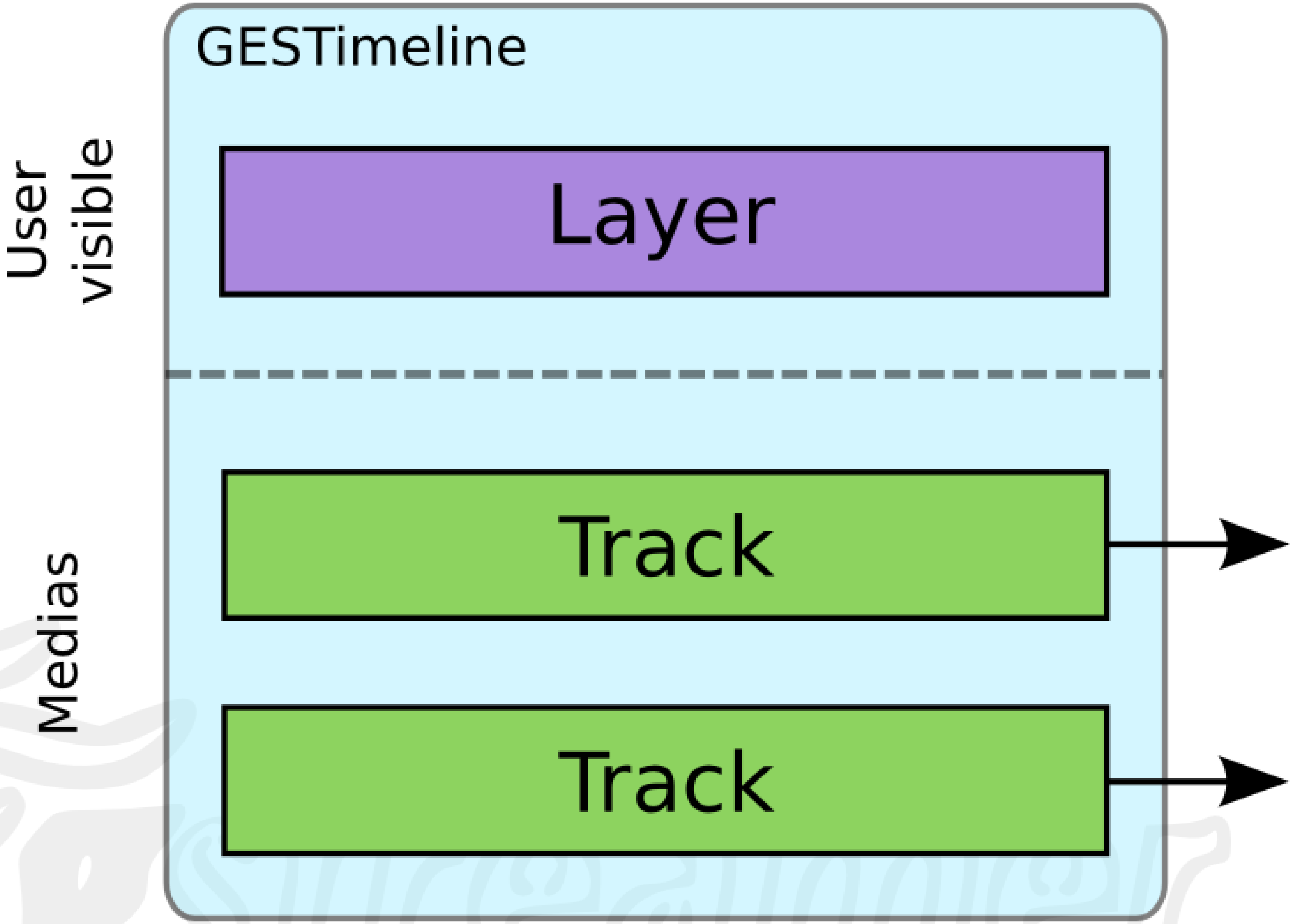
Gstreamer Editing Services

- Funded by Nokia
- LGPL
- 12 KLOC
- C/GObject, based on GStreamer and GnonLin.
- Examples, unit tests, API documentation
- High-level API
- Brandon Lewis co-developer
- Meego 1.3
- git.collabora.co.uk user/edward/gst-editing-services

GESTimeline

- Central object
- Controls Layers and Tracks
- Is a GstBin
- Save/Load

GESTimeline



GESTimelinePipeline

- GstPipeline (like playbin2)
- Takes a GESTimeline
- Playback/Preview *(autosinks)*
- Rendering *(encodebin)*
- Thumbnailing/Screenshot

GESTimelineLayer

- Takes «*natural*» objects
  - Files (Video, Audio, Images,..)
  - Transitions
  - Credits, Titles
  - ...
- Most user-centric part of the Timeline
- Can add more (ex: above:overlay, under:soundtrack)
- Media-agnostic

GESTimelineObject

- Basic properties
  - *Start* (When does it go ?)
  - *Duration* (For how long ?)
  - *In-point* (offset in the object)
  - *Priority* (Precedence over other objects)
- Creates and controls Track object(s)
- Base classes for Sources, Transitions, Overlays
- Create your own TimelineObject
  - Templates

Available TimelineObjects

- GESTimelineFileSource
  - Video, Audio, Picture...
  - Will figure out duration on its own
- GESTransition
  - Crossfade, most SMPTE transitions
  - Audio also

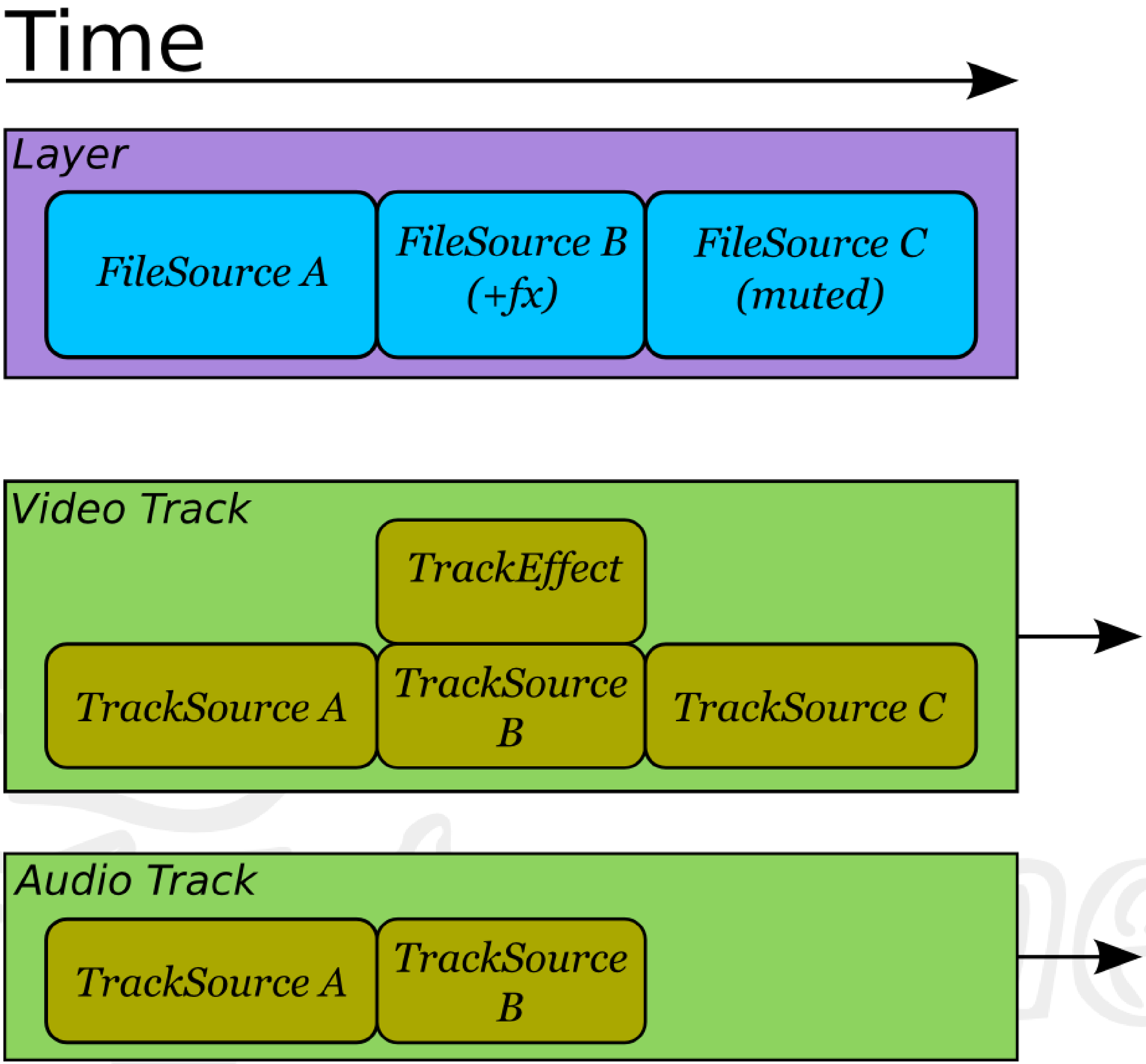
GESSimpleTimelineLayer

- List-based API
- Only care about the ordering and duration
- Takes care of adjusting the time position of all objects

GESTrack

- One per media output (Audio, Video, Subtitle, ...)
- Control what media is outputted
  - Raw Audio/Video...
  - ... or already encoded data
- Only set the Track(s) you want on the Timeline
  - Ex : video-only render/playback

Layer/Track interaction



GESTrackObject

- Produce/Modify the media
- GnlObject under the hood
- Essentially a GstBin
    - You can put anything you want in it

GESFormatter

- Timeline load/save (serialization)
- Create your own subclass

GstDiscoverer

- Get information about a URI
- Audio ? Video ?
- Duration ?
- Tags ?
- Codec ? Media properties (width/height...)
- Gst-plugins-base 0.10.31
- Used by GES if needed

libgstprofile and encodebin

- Make rendering as easy as playback
- Long standing problem
- GstEncodingProfile
  - Describe streams and not elements
- Encodebin element
  - (dynamic) sink pads based on profile
  - Can do passthrough
  - Conversion elements
- Proposed for gst-plugins-base
- Bugzilla #627476

gst_video_convert_frame

- Convert a video GstBuffer to any format
- Backported from playback plugin
- Added encoding capabilities (to images)

-
  ```
  GstBuffer* gst_video_convert_frame(
          GstBuffer *buf,
          const GstCaps *to_caps,
          GstClockTime timeout,
          GError **error)
  ```
- Gst-plugins-base 0.10.31

GstElementFactoryList

- Backported and improved from playback plugin
- « Get all factories of a certain <type> [, that can handle <media> [, in a certain <direction>]] »
  - Ex : Available video fx, encoders,...
- Gstreamer core 0.10.31

Lessons learnt

- Codecs:
  - Not the obvious cpu bottleneck
  - *GstSegment handling is not an option*
- Elements:
  - QoS for best end-user experience
- **Editing brings complex pipelines !**
  - Optimisation in core (caps nego, data passing, ...)
- Avoid memcpy (videoscale add-border)
- ORC ORC ORC !

Ideas / Improvements

- GnlComposition scheduling mode
  - Always ready (high mem, lowest lat)
  - Neighbour ready (med mem, med lat) + Using QoS
  - On demand (lowest mem, high lat)
- Single instance HW accelerated decoders
- pre-render/cache on-demand
- Proxy support
  - Of complex operations, sources... and timelines

Thankyou !

Any Questions ?