Implementing a Vulkan decoder

Víctor Jáquez
Igalia
API Overview

- Semi-stateless
- Low level
- Fine grained
  - Each driver implementation can operate differently
  -Verbose
- Big specification
Stream Parameters Sets

- Create a session parameters object
- Add new parameters
- Destroy session parameters object

This approach is very different from current parsers.
Picture references (DPB)

- References and output coincide
- Distinct references and output sets
  - references are a set of images
  - references are in a single multi-layered image

General use implementation should manage each case.
Bitstream buffers

- Upload **all** slices
  - Extra memory copies
  - VK_EXT_external_memory_host
- Prepend codec start-code
Multiplanar Images

- `VK_FORMAT_G8_B8R8_2PLANE_420_UNORM → NV12`
  - YUV420 2 planes in single image
- Currently, multiplane images are emulated with separate images per-plane

GStreamer format mapping issue:
2 Vulkan image layouts = 1 GStreamer format
GStreamer Capabilities template

- At element registering time
  - Query all possible decoding devices
  - Query all profiles each device is capable
  - Query all color formats each device/profile can output

```c
'pad Templates:
    SINK template: 'sink'
    Availability: Always
    Capabilities:
      video/x-h264
        profile: { (string)progressive-high, (string)constrained-high, (string)high, (string)constrained-baseline, (string)main, (string)baseline }
        width: [ 1, 4096 ]
        height: [ 1, 4096 ]
        alignment: au
        stream-format: { (string)avc, (string)avc3, (string)byte-stream }

    SRC template: 'src'
    Availability: Always
    Capabilities:
      video/x-raw
        width: [ 1, 4096 ]
        height: [ 1, 4096 ]
        format: NV12
        video/x-raw(memory:VAMemory)
        width: [ 1, 4096 ]
        height: [ 1, 4096 ]
```