Goals

- Cross-platform method to discover devices
  - Initially GPUs, but potentially other media devices
- Provide a platform-agnostic way to bootstrap an EGLDisplay
- Get out of the way of new window systems
  - Driver vendors shouldn’t be in this debate, or in the way
Goals (Cont.)

• Make Interop inside/outside EGL easier
• Need some way to identify devices across APIs

• Easy to adopt/Backwards compatible/Incremental change
• Don't completely ignore existing EGL APIs
• Don't try to do everything at once
Proposed Solution

- Introduce new Top-level EGL object above EGLDisplay
  - EGLDevice
- Add an EGL API to enumerate all devices on a system
- Allow creation of EGLDisplay directly from an EGLDevice
  - EGLDevice "platform" extension
- Introduce API to query the EGLDevice of an EGLDisplay
  - Makes it easy to use EGLDevice in existing libs/apps
How Do I Render to EGLDevice?

• Create an EGLDisplay from EGLDevice
• Create an EGLContext from EGLDisplay
• Two options:
  • MakeCurrent(ctx, EGL_NO_SURFACE)
  • Create an FBO
• OR
• Create an EGLStream producer surface
• MakeCurrent(cts, eglStreamSurf)
What About Display?

- A separate extension
- EGL "Output" extension enumerates outputs on a device
- A new modesetting API?!?
- Maybe
- Proposed workflow is:
  - Initialize EGLOutput on an EGLDisplay
  - Bind consumer end of EGLStream to EGLOutput object
  - SwapBuffers on stream producer surface consumed by EGLOutput
Allow New EGL native platforms

• ... Without editing EGL implementation
• Not directly related, but the last piece missing
• Add a "hook" system to "legacy" platform functions
  eglGetDisplay(), eglCreate[Window, Pixmap]Surface()
• Redirect them back to native platform libraries
• If native platform is based on EGLDevice, recurse back into EGL library and use EGLStream to implement surfaces.