EGL External platform API

- Motivation
- Overview & Structure
- Interactions with libEGL

A practical use case: Wayland

Conclusions
MOTIVATION

Decouple platform drivers from EGL drivers such that anyone writing a new window system can add a platform driver to integrate with existing EGL hardware drivers.

No vendor agreement required to support not-well-established window systems.

Platform code can be open source even if EGL drivers are not (less closed source is a win).

One common platform implementation improves consistency.
OVERVIEW

Specification of an API for writing EGL platforms and their interactions with modern window systems.

External platforms use application-facing EGL functions...

...leveraging config selection, context creation, and rendering support from lower-level EGL platform implementations (e.g. GBM, EGLDevice), ...

...or even other external platforms (e.g. X11, Wayland, Android).
EGL implementation $\rightarrow$ EGL external platform APIs:

- Pure EGL hooks
- Derivatives of EGL functions
- External object validation functions
- Handle translation

EGL external platform $\rightarrow$ EGL implementation APIs:

- Callbacks
STRUCTURE

Pure EGL hooks

Direct replacement of application-facing EGL functions for resource management.

Examples:

- eglGetPlatformDisplay()
- eglCreatePlatformWindowSurface()
- eglSwapBuffers()
Replace sub-parts of application-facing EGL functions.

An example of these is `queryString()` which takes custom ‘name’ tokens to retrieve, for instance, sub-strings of the extensions string.

The external platform manager can use each platform’s extension sub-string to compose the full string provided to applications.
FUNCTIONS SUCH AS `eglGetDisplay(<native_dpy>)`,
`eglCreateImage(EGL_WAYLAND_BUFFER_WL)`, OR
`eglCreateStreamAttrib(EGL_WAYLAND_EGLSTREAM_WL)` REQUIRE HELPER FUNCTIONS
TO DETERMINE WHAT EXTERNAL PLATFORM SHOULD HANDLE THOSE CALLS.

EXAMPLES:

- `isValidNativeDisplay(<native_dpy>)`
- `areStreamAttribsExternal(<attribs>)`
Non-externally implemented EGL functions will only understand internal EGL handles.

The external API defines the `getInternalHandle()` function so the internal EGL handle of an EGL external resource can be retrieved and passed along to internal functions.
For those operations requiring non-application-facing EGL paths to work, EGL implementations are allowed to register callbacks with the external platform implementations.

An example of these is setting EGL error codes to be queried by the application in case of failure in the external platform code.

By calling into `registerCallback(EXTERNAL_CALLBACK_SET_ERROR)`, an EGL implementation will let the external platform know the function to be called to set EGL error codes.
Discovery and registration of available EGL external platforms is libEGL’s responsibility.

A portable and fully configurable discovery mechanism is advisable (e.g. JSON loader).

`loadExternalPlatform(<major>,<minor>)` function:

- Initial libEGL → EGL external platform interaction
- Fills an EGL external platform exports table
- Lets libEGL select what API version to use
INTERACTIONS WITH LIBEGL

EGL calls dispatch

Application-facing EGL APIs

Application

libEGL
(keeps track of what handle belongs to what external platform)

Driver internal APIs

Driver internals

Application-facing EGL APIs

EGL external platform APIs

EGL external platform

EGL external callbacks
A PRACTICAL USE CASE: WAYLAND
NVIDIA’s Wayland platform is implemented on top of the EGLDevice and EGLStream families of extensions.

The simplest application work flow looks like:

- `eglGetDisplay()`
- `eglCreateWindowSurface()`
- `eglSwapBuffers()`

Let’s see how the three functions above are implemented in the Wayland external platform and backed by EGLDevice and EGLStream operations.
WAYLAND EXTERNAL PLATFORM

eglGetDisplay()

(*) For simplicity, wayland communication with the server is left out of the picture
WAYLAND EXTERNAL PLATFORM

eglCreateWindowSurface()

Application

eglCreateWindowSurface(dpy, native_win)

dpy->External::createPlatformWindowSurface(dpy, native_win)

libEGL

eglCreateStream(dpy->egl_dpy)

eglCreateStreamProducerSurface(dpy->egl_dpy, egl_stream)

eglGetStreamFileDescriptor(dpy->egl_dpy, egl_stream)

Driver internals

eglCreateStream(egl_dpy)
eglGetStreamFileDescriptor(egl_dpy, egl_stream)
eglCreateStreamProducerSurface(egl_dpy, egl_stream)

Wayland external platform*

(*) For simplicity, wayland communication with the server is left out of the picture
WAYLAND EXTERNAL PLATFORM

eglSwapBuffers()

Application

eglSwapBuffers(dpy, surf)

libEGL

surf->External::swapBuffers(dpy, surf)

eglSwapBuffers(dpy->egl_dpy, surf->egl_prod_surf)

Driver internals

eglSwapBuffers(egl_dpy, egl_prod_surf)

Wayland external platform*

(*) For simplicity, wayland communication with the server is left out of the picture

libEGL

eglSwapBuffers(dpy, surf)

Driver internals

eglSwapBuffers(egl_dpy, egl_prod_surf)

Wayland external platform*

(*) For simplicity, wayland communication with the server is left out of the picture
CONCLUSIONS
CONCLUSIONS

Introduced an EGL external platform API to **decouple platform drivers** from EGL drivers.

Successfully added **Wayland support as an external platform** on top of the EGLDevice and EGLStream families of extensions.

We will soon **open source** both the EGL External platform API and the Wayland implementation.
EXTRA: EGL EXTERNAL API

struct ExternalEglExports {
    registerCallback;
    isValidNativeDisplay;
    bindDisplays;
    unbindDisplays;
    getPlatformDisplay;
    initialize;
    terminate;
    chooseConfig;
   getConfigAttrib;

    [...]
};