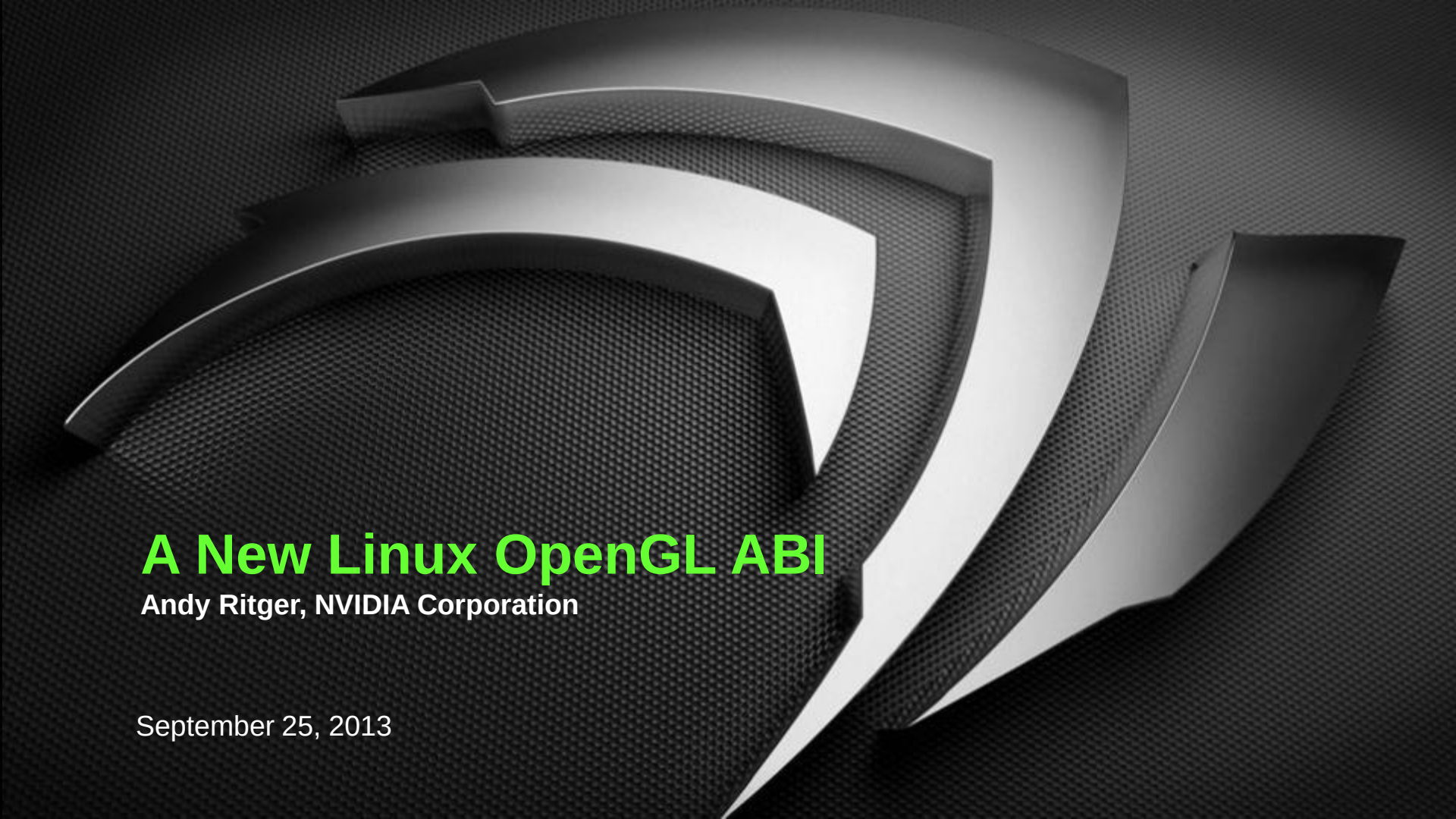

A New Linux OpenGL ABI

Andy Ritger, NVIDIA Corporation

September 25, 2013

History

The current Linux OpenGL ABI states:

- “libGL.so.1 ... must export all OpenGL 1.2, ..., GLX 1.3, and ARB_multitexture entry points statically.”

This is a bit antiquated: EGL has become a compelling alternative to GLX. It would be nice for OpenGL entry points to be provided separately from GLX or EGL. It would also be nice to guarantee more modern OpenGL entry points.

- “libGL.so.1 ... must be located in /usr/lib.”

We don’t have a good standard mechanism for multiple OpenGL implementations to coexist on the filesystem. In practice, vendors frequently stomp on each at driver install time.

Goals

- Define new ABI between applications and OpenGL libraries:
  - Window System libraries: EGL, GLX
  - Client API libraries: OpenGL, OpenGL ES
- Allow multiple vendor implementations to co-exist on the file system.
- Allow multiple vendor implementations to co-exist within the same process.
- Allow vendors to provide extensions and new OpenGL versions on their schedule.
- Preserve backwards compatibility.

New Library Organization: Summary

- Vendor-neutral Client API Libraries:
  - libOpenGL.so.1
  - libGLESv1_CM.so.1
  - libGLESv2.so.1
- Vendor-neutral window system libraries:
  - libGLX.so.1
  - libEGL.so.1
- Vendor-specific libraries:
  - libGLX_${VENDOR}.so.1
  - libEGL_${VENDOR}.so.1

New Library Organization: Vendor-Neutral Client API Libraries

- libOpenGL.so.1
  - Provides symbols for OpenGL 4.4 (Core and Compatibility Profiles).
  - Vendors can provide additional OpenGL entry points, through {egl,glX}GetProcAddress.
  - No EGL or GLX entry points provided by this library; should be used with lib{GLX,EGL}.so.1.
- libGLESv1_CM.so.1
  - Provides symbols for all OpenGL ES 1 common profile entry points.
  - No EGL or GLX entry points provided by this library; should be used with lib{GLX,EGL}.so.1.
- libGLESv2.so.1
  - Provides symbols for all OpenGL ES 2 and 3 entry points.
  - No EGL or GLX entry points provided by this library; should be used with lib{GLX,EGL}.so.1.

New Library Organization: Vendor-Neutral Window System Libraries

- libEGL.so.1
  - Provides symbols for all EGL 1.4 entry points.
  - Loads and dispatches to one or more vendor libraries.
- libGLX.so.1
  - Provides symbols for all GLX 1.4 entry points.
  - Provides symbols for the GLX_ARB_create_context extension.
  - Loads and dispatches to one or more vendor libraries.

New Library Organization: Vendor-Specific Libraries

- lib{EGL,GLX}_${VENDOR}.so.1
  - Provides initialization function that lib{EGL,GLX}.so.1 calls.
  - Pulls in the vendor's implementation of all the client APIs it supports.
  - Registers with the appropriate Client API library at MakeCurrent time.
  - Must not export symbol names that collide with:
    - EGL (^egl.*),
    - GLX (^glX.*), or
    - OpenGL (^gl.*).

Dispatching to Vendor Implementations

- The vendor-neutral libraries need to dispatch each entry point to a vendor.
- Easy for Client APIs:
  - MakeCurrent defines the vendor to use with the thread.
- Slightly harder for Window System APIs:
  - Many EGL,GLX entry points imply a vendor through their arguments.
  - Some EGL,GLX entry points dispatch based on the current context. E.g., eglWaitGL.
  - Some EGL,GLX entry points return current API state. E.g., glXGetCurrentContext.

  In nearly all cases, the vendor for a window system entry point can be inferred.
- We are not trying to address server-side GLX (yet), so in practice GLX can only have one vendor at a time, per X server, for now.

Example: libGLX.so.1 and libOpenGL.so.1

a) Application calls any GLX entry point.
b) libGLX.so.1 queries the X server, to map X screen to vendor.
c) libGLX.so.1 loads and dispatches to libGLX_${VENDOR}.so.
d) During glxMakeCurrent, libGLX_${VENDOR}.so registers with libOpenGL.so.1; sets up dispatch table.
e) Application calls OpenGL entry point.
f) libOpenGL.so.1 jumps through dispatch table to OpenGL implementation registered by libGLX_${VENDOR}.so.

Example: libEGL.so.1 and libOpenGL.so.1

a) Application calls eglInitialize.
b) libEGL.so.1 uses configuration magic to select, load, and dispatch to libEGL_${VENDOR}.so.
c) During eglMakeCurrent, libEGL_${VENDOR}.so registers with libOpenGL.so.1; sets up dispatch table.
d) Application calls OpenGL entry point.
e) libOpenGL.so.1 jumps through dispatch table to OpenGL implementation registered by libEGL_${VENDOR}.so.

Backwards Compatibility

- There will be a libGL.so.1 provided with the vendor-neutral libraries.
- This exports all symbols from all current vendors' libGL.so.1's.
- For symbols provided by libGLX.so.1 or libOpenGL.so.1:
  - use ELF DT_FILTER to resolve libGL.so symbols with libGLX.so.1, libOpenGL.so.1.
- For symbols not provided by libGLX.so.1 or libOpenGL.so.1:
  - use libGLX.so.1's glXGetProcAddress to call from libGL.so.1 to libGLX.so.1, libOpenGL.so.1.
- Existing applications *should* be unaffected.

Distribution

- Hopefully vendor-neutral libraries require very little maintenance once implemented.
- Once in maintenance mode, contribute to Khronos for hosting.
- Linux distributors would distribute the vendor-neutral libraries.
- Vendor libraries packaged separately from the vendor-neutral libraries.

NVIDIA's Migration Plan

- Update NVIDIA OpenGL implementation to provide lib{EGL,GLX}_nvidia.so
- Rely on presence of vendor-neutral libraries.
- Encourage Linux distributions to package the vendor-neutral libraries, once they are ready.
- If the Linux distribution doesn't provide the vendor-neutral libraries, NVIDIA's installer would install its own copy of the vendor-neutral libraries.

Challenges: GetProcAddress

Problem:

- GetProcAddress is context (i.e., vendor) independent.
- lib{EGL,GLX}.so cannot dispatch GetProcAddress calls to a vendor implementation.

Solution (sort of):

- If the function name is a function GetProcAddress already knows, return its pointer.
- Otherwise, online-generate a new entry point stub for every unique string GetProcAddress receives.
- Map these entry point stubs to vendor functions at MakeCurrent time.
- Works for Client API functions, because they are per-context.
- Not good enough for window system functions dispatched by function arguments, rather than current context.

Challenges: Mapping GLX Objects to Vendors

Problem:

- To dispatch some GLX entry points to a vendor, libGLX.so.1 needs to use the GLX object argument (e.g., GLXContext, GLXDrawable) to correlate to a screen and then to the vendor for that screen.
- libGLX.so.1 can use client-side tracking of which objects are allocated on which screen.
- GLXDrawables are XIDs that could be created by another process, so libGLX.so may not have enough information to dispatch on its own.

Solution (mostly):

- Add an X server request to map from XID to screen.
- Theoretically racy: the XID could be destroyed and reallocated for an object on another X screen.

Implementation Status

- Rough Implementation available here: http://github.com/NVIDIA/libglvnd
- libGLX.so.1 and libOpenGL.so.1 implemented.
- libEGL.so.1, libGLESv1_CM.so.1, libGLESv2.so.1 not yet implemented.
- Compatibility libGL.so.1 partly implemented.
- Uses Mesa's glapi code for thread local storage and entry point stub generation.
- Will likely use some of Mesa's EGL implementation for libEGL.so.1.
- A lot of implementation details still to be worked out.
- Next step: plug NVIDIA's OpenGL implementation into it.
  - Should help identify problems with the implementation so far.
  - Based on our experience doing this, we could write a brief "porting guide" that might be helpful for other vendors.
- Feedback, code review, etc very welcome.

Miscellaneous

- FAQ: Why isn’t there a libOpenGL_${VENDOR}.so?
- Answer: The only DSOs the vendor-neutral libraries need to look for are lib{EGL,GLX}_${VENDOR}.so. Vendors will likely want a common DSO for the OpenGL implementation, shared by GLX and EGL, but that is an implementation detail.

- Thanks to Brian Nguyen for most of the implementation work, so far.

References

- These slides: https://github.com/aritger/linux-opengl-abi-proposal/blob/master/presentation-xdc2013.pdf
- Proposal: https://github.com/aritger/linux-opengl-abi-proposal/blob/master/linux-opengl-abi-proposal.txt
- Current implementation: http://github.com/NVIDIA/libglvnd
- Email thread for discussion of the implementation: http://lists.freedesktop.org/archives/mesa-dev/2013-August/044021.html

Questions