

Towards Open Source 3D Acceleration For Nvidia Cards

Stéphane Marchesin
marchesin@icps.u-strasbg.fr



Introduction

- Dependence on proprietary drivers
 - ◆ Future window systems will be layered upon OpenGL
 - ◆ 3D applications
 - ◆ 3D heavily used in games
- Proprietary Nvidia drivers
 - ◆ No luck on non-x86 hardware
 - ◆ Inability to fix bugs
 - ◆ Long time support ?

Nvidia : the hardware

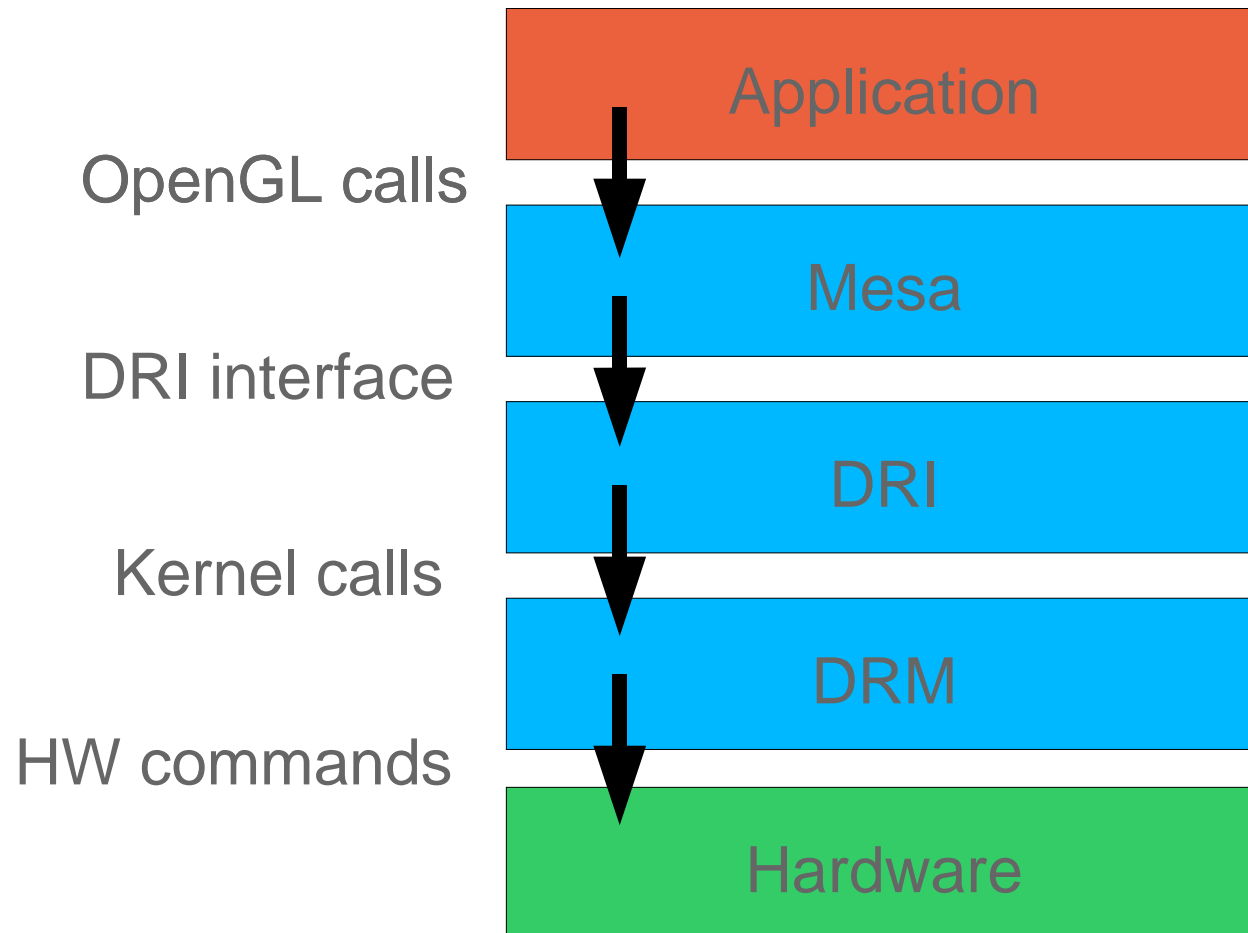
- GeForce 6x00 (NV40)
 - ◆ Multiple hardware contexts (since NV3 !)
 - In the form of multiple command fifos
 - At least 8 contexts
 - ◆ OpenGL 2.0 hardware
 - Powerful
 - Complex
 - ◆ No documentation available
 - Source code used to be available (up to NV5)
 - The “nv” DDX (all cards)
 - The Utah GLX driver (up to NV18)
 - The BeOS 3D driver (up to NV18)

The DRI/DRM model

- DRM module protects access to the card
 - ◆ In-kernel
 - ◆ Low footprint
 - ◆ Has to check each command for security
 - Can be costly
 - State tracking to avoid useless calls
 - Complex implementation
- DRI module makes most of the work
 - ◆ User space
 - ◆ Plugs into Mesa
 - ◆ Builds command packets
 - ◆ Makes kernel calls to submit commands to the DRM

The DRI/DRM model

- DRM-managed command submission

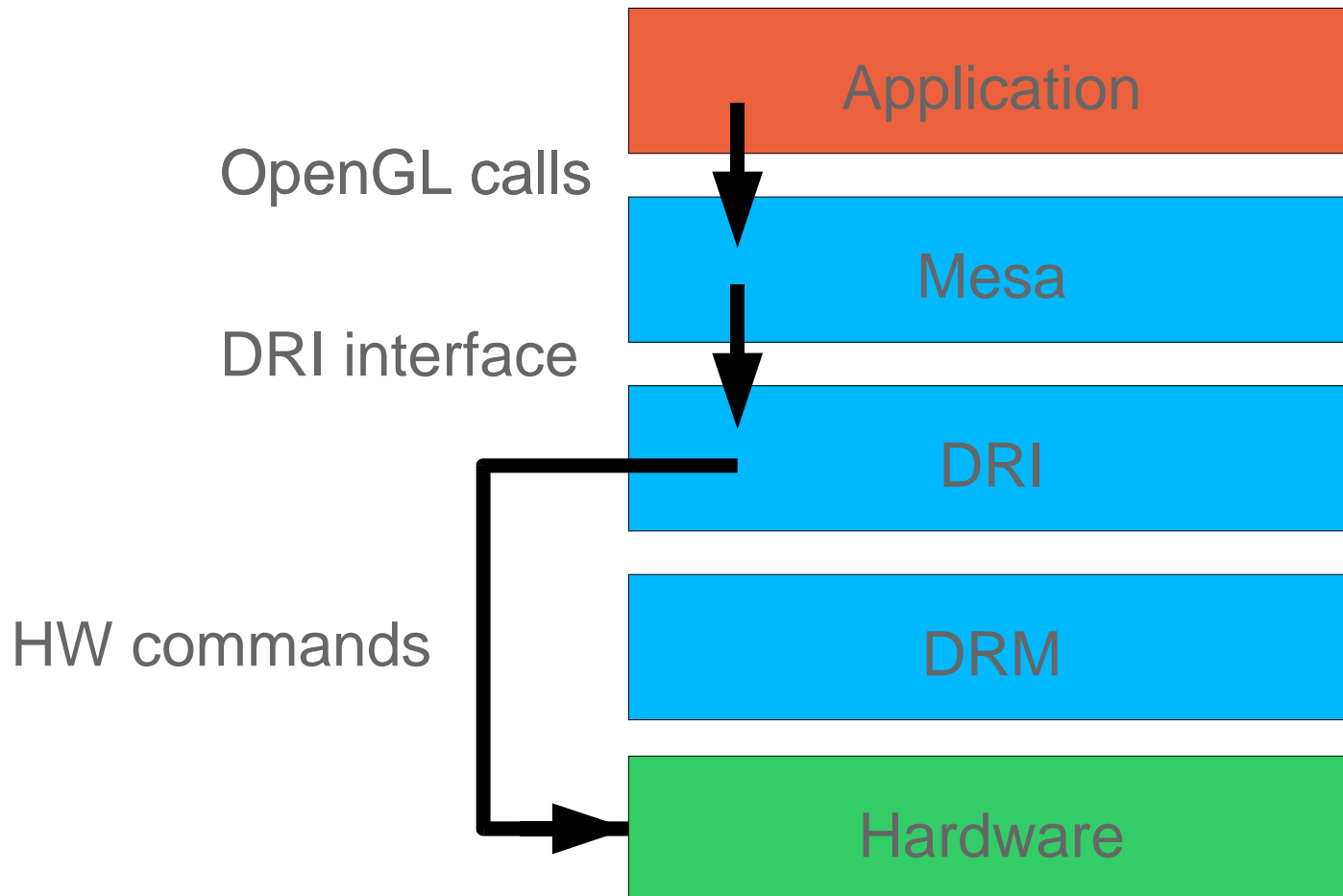


Relaxing the DRI/DRM model

- Nvidia hardware has multiple fifos
 - ◆ DRM maps one fifo per client (RW)
 - ◆ DRI client then has exclusive use of its fifo
 - Can fill it as it likes
 - Full OpenGL primitive submission in user-space
 - No need for context switches
 - No need for (some of the) mutex locks
 - Of course, other things still need to be checked (DMA accesses)
 - DRM update not always needed for new functionality

Relaxing the DRI/DRM model

- Full user-space command submission



- The DRM initializes and setups the registers
- Initialize the multiple rendering contexts
- Setups a fifo when requested by a DRI client and maps the fifo to the client
- That's about it (lazy guy)

- The DRI initializes, maps the fifo from the DRM
- Primitive submission can then work without the DRM's help
- Full fifo control happens in user space
 - ◆ No complicated code for context switching and tracking
 - ◆ No need for a kernel call
- Emitting primitives
 - ◆ DRI emits primitives to the fifo
 - ◆ DRI flushes the fifo

- Functionality needs to be added to the DDX
 - ◆ Back/depth buffers
 - ◆ Swapbuffers
 - ◆ Cliprects
 - ◆ ...
- On top of the EXA patch
- DDX is hardcoded to use context 0
 - ◆ Always reserve this context

- Need to figure out functionality for NV20 and later cards

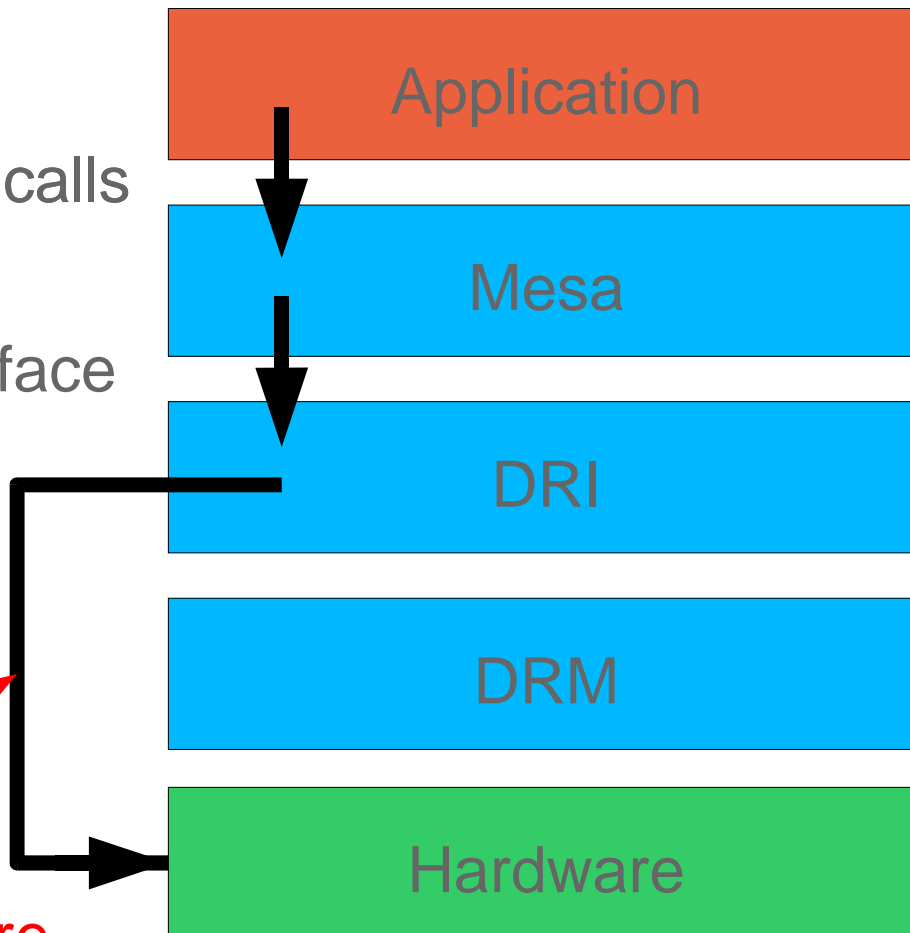
- ◆ How ?

OpenGL calls

DRI interface

HW commands

HW Commands are here !



■ Solution

- ◆ Create an OpenGL process
- ◆ Find the fifo among the mappings
- ◆ Dump the fifo & registers contents
- ◆ Do something with the graphics pipeline
 - `glClear()`
 - `glVertex()`
 - ...
- ◆ Compare the fifo & registers with the previous state
- ◆ Deduce functionality

Reverse engineering applied

■ Working with vertices

- ◆ Vertex submission
 - Send 1 vertex
 - Send 2 vertices
 - ...
 - Send X vertices
 - Compare the results
 - Deduce how to submit vertices
- ◆ Vertex description
 - Send color vertices
 - Send color+lighting vertices
 - Send textured vertices
 - ...
 - Compare

Show me the code !

- (Not yet working) code : <http://nouveau.sf.net>
- Assumes NV40

Conclusions

- Lots of work left
 - ◆ Adapt the DRM to do client-exclusive mappings
 - ◆ Contexts >0 need initialization code
 - ◆ Add back/depth buffers
 - Needs a memory manager
 - ◆ Textures (needs some DMA support)
 - Textures compete with pixmapes for video ram
 - Once again, memory manager
 - ◆ NV40 is being looked at
 - Explore other chips
 - Keep a unified driver
 - ◆ Reverse engineering works
 - But not all HW information can be found that way

Thanks !