State of the Art Debugging and Tuning Graphics Applications

Ian Romanick <ian.d.romanick@intel.com>
18-September-2013
Agenda

• Windows vs. Linux debugging graphics applications
• Windows vs. Linux performance tuning applications
• What app developers are really doing
• How can we enable them?
Debugging

- Multiple closed-source tools
  - AMD GPU PerfStudio 2
  - NVIDIA® Nsight™
  - Intel® Graphics Performance Analyzers
  - Visual Studio Graphics Debugger

- Driver assisted debugging
  - GL_ARB_debug_output

- Some open-source tools
  - apitrace
  - BuGLE (no longer maintained)

- Few closed-source tools
  - NVIDIA® Nsight™
  - Gremedy gDebugger (no longer maintained)

- Driver assisted debugging
  - GL_ARB_debug_output
  - gdb breakpoints in the driver
Performance Tuning

- Multiple closed-source tools
  - Same set of vendor tools
  - GPUView
  - PIX (obsolete)

- Driver assisted performance monitoring
  - GL_ARB_timer_query
  - GL_AMD_performance_monitor

- Few vendor tools
  - NVIDIA® Nsight™
  - intel_gpu_top (open-source)

- Some open-source tools
  - fips
  - apitrace (sort of)

- Driver assisted performance monitoring
  - GL_ARB_timer_query
  - GL_AMD_performance_monitor
  - Gallium performance HUD
  - INTEL_DEBUG=perf,shader_time
  - Also GL_ARB_debug_output
Universal Problems

• Many tools are vendor specific
  • “I really like [vendor X's tool], but I wish it worked better on [vendor Y's hardware].”

• Tools use driver back-channels to get low-level performance data

• Developers work on one GPU, then “port” to the next

• Many tools are DirectX specific
  • “I want PIX for OpenGL.”

• Many tools give narrow view of the system.
  • What is the compositor doing to my apps performance?
  • When am I blocked on the X server?
  • etc.
  • Major issue for browser vendors
What do developers really do?

- Roll their own tools
  - Data collection in the application with and external, post-hoc visualization tool
  - Data collection and visualization in the application
- Both routes involve ugly choices
  - Manual insertion of trace points in the application
    - Think fancy `rdtsc()` calls everywhere...
  - Valve presented about Telemetry at SIGGRAPH 2012
  - Use generic, imprecise collection methods that work on all hardware or...
  - Use vendor-specific, detailed collection methods that only work on one vendor
The $64,000 question...

• Can we provide a set of interfaces, probably from the kernel, that:
  • Provides finer grained data than is available from GL_ARB_timer_query about the execution of commands on the GPU.
  • Provides time information of command submission and completion relative to vblank, CPU profiling events, etc.
  • Provides all that data at a system level with semantic information
    • This block of time was your call to glDrawArrays
    • This block of time was the compositor doing "stuff"
    • This block of time was your XRender request
    • etc.
The $64,000 question...

• And the hard parts...
  • Doesn't leak information in a way that compromises security
  • Allow closed-source drivers to expose these interfaces
    • ...and not garner too much rage from the maintainers