Customized graphical driver for profiling GL state On Ubuntu

* Team members:

Ryan Polley, Pranjal Abhyankar, Vivek Mishra, Wanxin Wang

* Project delivery:

An open source driver for logging, profiling GL state

* Hardware requirement:

Graphic card installed that supports 3D acceleration, graphics driver supports DX11 (Windows) or OpenGL 4.0 (Linux, Mac)

* Introduction:

Inspired by: <https://fgiesen.wordpress.com/2011/07/01/a-trip-through-the-graphics-pipeline-2011-part-1/>, we would work on inspecting graphics chain on operating system, from high-level graphics application, down to the graphical driver in the operating system kernel. Mesa + Linux (<https://en.wikipedia.org/wiki/Free_and_open-source_graphics_device_driver>) gives feasibility for it, since there is an unbroken chain of open source software between the OpenGL layer and graphics card instructions, which allows for the code that does it to be easily modified for profiling. While through studying the code in different levels of graphics chain, we could log the graphics system calls (OpenGL API, EGL API…), as well as the activites of the Mesa driver, profile for OpenGL calls, and customize the components in chain to yield a graphical output from different levels.

* Approach

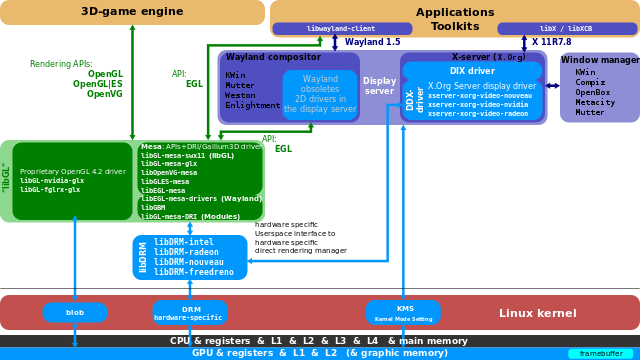


Figure1: The graphics stack on Linux

The inspection would be along the following paths:

* + OpenGL Application->libGL + MESA driver->libDRAM
  + EGL application(2D GUI)-> libGL + MESA driver->libDRAM
  + (Optional) 3D-game engine-> libGL + MESA driver->libDRAM

One challenge is that a significant portion of extra work results from supporting heterogeneous environments: if an OpenGL application is ported from windows to Ubuntu Linux running on VMware, or to Ubuntu Linux running on a bare metal, the differences between the Mesa drivers increases the likelihood of bugs, requiring more caution overall. We plan to tackle this through small but rapid incremental steps which would allow us to consistently make progress without introducing errors. Additionally, the build environment for this would be complex, and would require some effort to setup. Another technically challenging aspect of the project would be that it would allow for execution tracing on the relatively high level of OpenGL code, but also on the level of the Mesa EGL and the libDRM code. Since the final level, libDRM, is so low level, emiting as output GPU machine code, understanding what it's doing and distilling that into information useful for profiling or debugging could prove itself an additional technical challenge.

* Comparison with existing OpenGL profiling tools

The mechanism of the existing OpenGL tool would work in following structure:



Figure2: Comparison to a generic profiling tool

From this structure, if the generic tool works with an incompatible driver(libGL), either can it not work with an error prompted, most complaints are of this issue, or more seriously, the profiling tool generates not accurate results and it undergoes unnoticed. Thus it is mostly recommended to bind to driver and tool together for profiling purpose, and our approach will work on both OpenGL application source code and libGL source code, we could more closely measure the logging and profiling introduced performance drawback, generate the result more countable. The existing tools give the options for logging with either GUI operations, or with configuration files. But our approach works on source code level, we have much more flexibility, we can select the portion of data we want to log in our preferred format, that would be easier for analyzing tools to analyze.

* Experiment

The original Mesa driver will be customized in order to generate trace log information of OpenGL state for the usage of OpenGL and EGL from graphical application, and yields graphical output, for example to add a water mark on the display for every graphical application that uses Customized driver, to profile hardware, and graphical driver information, to compare performance of customized driver + Ubuntu with performance of other platforms.

* Milestones
  + Port OpenGL demo code from Windows to Ubuntu (VMware version)
    - Wanxin Wang ports it
    - The rest of team verifies it
  + Port OpenGL demo code from Windows to Ubuntu (Bare metal version)
    - Ryan Polley, Vivek Mishra
  + Upgrade Ubuntu (VMware guest) graphics driver with latest Mesa solution
    - Wanxin Wang, Pranjal Abhyankar
  + Add FPS(frame per second), and other real time state report
    - Ryan Polley, Pranjal Abhyankar, Vivek Mishra, Wanxin Wang
  + Midterm report
    - Ryan Polley, Pranjal Abhyankar, Vivek Mishra, Wanxin Wang
  + Add OpenGL state logging functionality to OpenGL APIs
    - Wanxin Wang, Pranjal Abhyankar, Vivek Mishra
  + Initiate EGL demo code
    - Ryan Polley
  + Add logging functionality to EGL APIs
    - Ryan Polley, Pranjal Abhyankar, Vivek Mishra, Wanxin Wang
  + Set up Ubuntu environment on a bare metal with latest customized driver solution
    - Ryan Polley, Pranjal Abhyankar, Vivek Mishra
  + Log function calls to libDRM (hardware specific interface in user space)
    - Ryan Polley, Pranjal Abhyankar, Vivek Mishra, Wanxin Wang
  + Port unreal engine game to the bare metal + customized driver solution (Optional)
    - Wanxin Wang
  + Final report
    - Ryan Polley, Pranjal Abhyankar, Vivek Mishra, Wanxin Wang