Gallium3D Components

Brian Paul
Nov 13, 2009
Virtual Machine Group
Status of Open Source Components

Gallium
* Graphics API Support
* Window System Bindings/APIs
* Device Drivers

Mesa
* Mesa status
Graphics API Support

“State Trackers” are the “front-ends” to Gallium which translate standard graphics APIs into Gallium calls.

Rendering APIs:
* OpenGL 2.1
* OpenGL ES 1.x
* OpenGL ES 2.x
* OpenVG

Cool thing: each new Gallium device driver will automatically support all those APIs (and more)!

Binding interfaces:
* GLX
* EGL
OpenGL 2.1 State Tracker

The Gallium OpenGL state tracker is really just core Mesa plus a “Mesa Gallium driver”.

The pieces include:

* Core Mesa: OpenGL API functions, context state management (src/mesa/main/)

* Mesa's GLSL compiler: (src/mesa/shader/)

* Vertex buffer object builder: translates OpenGL drawing commands into a uniform Vertex Buffer Object (VBO) representation (src/mesa/vbo/)

* Mesa/Gallium device driver: a Mesa device driver that targets gallium. Implements all the ctx->Driver.Foobar() hooks. (src/mesa/state_tracker/, but may be moved someday)
Status of Open Source Components

Conventional Mesa driver stack

- Core Mesa + GLSL
  - TNL, swrast, etc
  - Mesa Device Driver
    - Somewhat large
    - Redundant code
    - Device specific:
      - r200/r300/etc
      - i945/i965
      - nouveau/sis/via

Mesa + Gallium

- Core Mesa + GLSL
  - Mesa state tracker
    (aka Mesa Gallium driver)
    - Fairly small
    - Device independent
  - Any Gallium driver
OpenGL ES and EGL State Trackers

* Support for OpenGL ES 1.0, 1.1 and OpenGL ES 2.0

* OpenGL ES implemented as a subset of Mesa

* EGL 1.0, 1.1, 1.2, 1.3 support

* Originally developed by Bob Ellison and Brian Paul at Tungsten Graphics

* Recently “taken over” by Chia-I Wu who's working on Android integration

* More details later in “OpenGL ES State Tracker Status”
OpenVG State Tracker

* Functional and nearly complete
* To be described in detail by Zack later.
Status of Open Source Components

Window System Binding Interfaces

GLX
* The libGL.so library
* Dynamically loads xxx_dri.so drivers which may be a conventional DRI driver or a gallium-based driver.
* Also, an emulated GLX library (doesn't depend on any X server support for GLX). Useful for testing/debugging.

EGL
* The window system interface for OpenGL ES
* Also supports full OpenGL and OpenVG
* Can load new EGL-based drivers or legacy drivers with the egl_glx shim (converts EGL API calls into GLX API calls)
Gallium Device Drivers

Softpipe: The “reference driver” for Gallium. Slow, but as correct as possible.

LLVMpipe: New, fast software driver which uses LLVM for run-time code generation (shaders triangle rendering, etc). Only supports x86 at this time.

I915: For Intel i915/i945 hardware. Pretty much complete

I965: Just started, spare-time project.

Cell: Software driver for the IBM/Sony/Toshiba Cell processor. Functional, but not complete. Not actively being developed anymore.


AMD R300: Under development by Corbin Simpson, Cooper Yuan, Nicolai Hahnle, et al.
Status of Open Source Components

Future Plans for Core Mesa

* New extensions and API support: OpenGL 3.x features, Geometry Shaders, new texture formats, new rendering commands, etc. (covered in more detail later)

* New, improved GLSL compiler (Michal Krol’s preprocessor + Ian Romanick’s yacc-based parser + LLVM)

* Code re-factoring: Partition files and functions according to API: OpenGL 1.x/2.x vs. OpenGL 3.x vs. OpenGL ES 1.x vs. OpenGL ES 2.x, etc. (create GL building blocks)

* Move some pieces to new directories: GLSL compiler, GPU program code

* Refine data structures: Ex: unify textures and renderbuffers and add texture buffer object support

* Remove unused driver hooks (GLSL-related functions, vertex arrays, etc)