

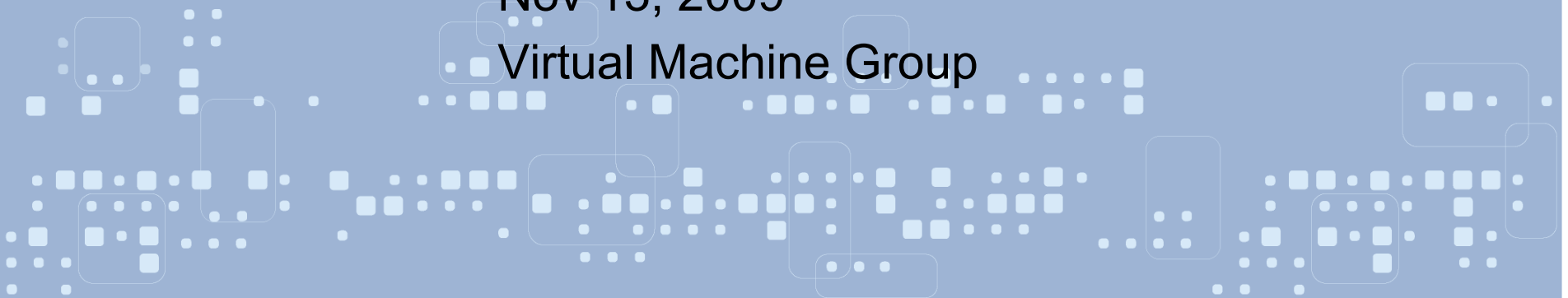


# OpenGL-ES State Tracker Status

Brian Paul

Nov 13, 2009

Virtual Machine Group



## OpenGL ES / EGL State Tracker Status

\* At Tungsten Graphics, Bob Ellison and Brian Paul implemented OpenGL ES 1.1 and 2.0 state trackers for Mesa/Gallium in 2008.

\* There were two possible implementation paths:

1. Write new state trackers from scratch, directly interfacing with Gallium.
2. Re-use a subset of Mesa plus the Mesa/Gallium state tracker/driver.

We went with option 2.

\* Despite OpenGL ES 1 and ES 2 being small subsets of OpenGL 2.1, there's still a lot of common code. For example: state management (rasterization, blending, Z, stencil), vertex array/buffer drawing commands, texture objects, GLSL compiler, etc.)

## OpenGL ES / EGL State Tracker Status

Changes in Mesa to support ES 1.1 and ES 2.0:

- \* Repartitioned some core Mesa code to make it more modular. For example, separate `glTexGen()` functions into a separate file which is omitted from the ES builds.
- \* Made assorted changes for OpenGL ES:
  - \* Point size arrays
  - \* `GL_BYTE` vertex arrays
  - \* Added some ES extensions like `GL_OES_query_matrix`, `GL_OES_draw_texture`, `GL_OES_compressed_paletted_texture`, etc.
  - \* GLSL changes, such as precision qualifiers
  - \* Enable Point Sprite mode by default
  - \* Different default value for buffer mapping state
  - \* Restricted parameters for `glRenderbufferStorage()`, etc.
- \* Very few changes to the Mesa/Gallium state tracker.

## OpenGL ES / EGL State Tracker Status

Modified the build to omit the non-ES Mesa source files:

- \* Created `src/gallium/state-trackers/es/` with `es1/` and `es2` subdirs.
- \* Symlink the re-used Mesa sources into those directories and build there.
- \* Produce `libGLESV1_CM.so` and `libGLESV2.so` libraries
- \* Since then, Chia-I Wu has been reworking OpenGL ES support. See the `opengl-es-v2` branch in Mesa git.

## OpenGL ES / EGL State Tracker Status

### EGL

- \* EGL is a window system interface for creating rendering contexts and binding them to drawing surfaces, similar to GLX, but without the window-system specifics of GLX.
- \* EGL may be used with OpenGL, OpenGL ES, OpenVG, etc.
- \* Mesa's implementation of EGL is very modular and flexible. More flexible than GL X/ libGL.
- \* libEGL.so does the following (see src/egl/main/):
  - \* Implements the egl API functions like eglCreateContext() and eglSwapBuffers().
  - \* Handles basic EGL surface, context, screen management.
  - \* Device driver loading.
- \* An “EGL device driver” may either be a real device driver or a “shim” that in-turn loads other drivers.

## OpenGL ES / EGL State Tracker Status

### EGL continued:

\* Example shim library: `egl_glx.so` is loaded by `libEGL` and it, in turn, loads a conventional DRI/GLX driver. Requires an X server plus GLX.

Alternately, full EGL drivers are assembled from building blocks:

\* Example hardware driver:

1. A gallium driver, such as `libi915.a` for the Intel 915/945 GPU
  2. A winsys, such as `libinteldrm.a`
  3. A state tracker, such as `libegldrm.a` (full-screen, X-less EGL)
- Combine the building blocks to create an EGL driver: `EGL_i915.so`

\* Example software driver:

1. A software gallium driver such as `softpipe` (or `LLVMpipe` or `cell`)
  2. A winsys such as `sw_winsys` (software-based buffers and fences)
  3. A state tracker, such as `egl_xlib` (render into X windows)
- Combine the building blocks to create `egl_softpipe.so`