Glamor Status Report

Keith Packard
Open Source Technology Center
Intel
keithp@keithp.com
What is Glamor?

- Glamor X Rendering helper
- Hardware independent
- Supports EGL and GLX
- Supports GL and GLES
Where Did Glamor Come From?

- Eric started it
  - December 2008
  - Goal of offering efficient hardware-independent X acceleration
- GL was pretty dire at the time
  - Lots of 1.x drivers
  - Lame 2.0 shader support
- Adopted by Zhigang Gong and Junyan He
  - April 2011
  - Goal of supporting SGX hardware without lots of custom code
- GL was a lot better
  - Widespread GL 2.x support
Glamor Status in mid 2013

- Mostly Complete X acceleration
  - Missing planemasks and a few other operations
- Structured like fb
  - Build simple function to draw one object
  - Layer with CPU-intensive code to deal with clipping and repeats
- Performance heavily limited by CPU cost in Glamor and OpenGL library
More Recent Glamor History

- Radeon stopped offering non-Glamor acceleration
- Re-adopted by Eric
  - August 2013
- Piled on by Keith
  - March 2014
Pixmaps in Glamor

- GL Textures limits generally smaller than X pixmap limits
- Tile textures to fill pixmap
- Dest is easy; just replicate rendering to each tile member
- Source requires some magic
  - Compute rectangle of dest covered by one source tile
  - Construct intermediate textures from multiple source tiles to eliminate seams in dest
X and Pixel Formats

- Pixmaps have no intrinsic color information. Just depth.
- Windows have a visual, which describes their pixel's RGB layout. Bits beyond those have no core protocol meaning.
- Render Pictures imbue pixels with color and alpha.
- Pixmaps (and even Windows) can have multiple Pictures with different PictFormats.
GL and Pixel Formats

- There are four “channels”, R, G, B and A.
- Textures have intrinsic channel information, but no depth or layout.
- Surfaces have channel information describing which channels they contain.
- Data transferred between the application and textures includes layout information.
- ARB_texture_swizzle lets you remap the channels (if present).
Matching X and GL formats

• PutImage/GetImage specify the X wire format to GL
• ARB_texture_swizzle can help with some image format changes.
• However, sometimes Glamor must reformat data with the CPU.
• Glamor doesn't currently do this correctly.
Fallbacks for Glamor

• What to do when GL actually doesn't work
• Download all pixmap textures to PBO
• Map, fallback to fb
• Upload PBO back to textures
• Can take bounding box to limit data transfer
Glamor for Core X

- Rewritten in mid 2014
  - Goal was to
- Eliminate CPU time spent in Glamor
- Use GPU for complete operations
Dynamic Shader Generation

• Fragments of GLSL for each phase of rendering
• Glued together and compiled at runtime
Rect Shader (GL)

```cpp
static const glamor_facet glamor_facet_polyfillrect_130 = {
    .name = "poly_fill_rect",
    .version = 130,
    .vs_vars = "attribute vec4 primitive;\n",
    .vs_exec = ("    vec2 pos = primitive.zw *
              vec2(gl_VertexID&1, (gl_VertexID&2)>>1);\n" GLAMOR_POS(gl_Position, (primitive.xy + pos))),
};
```
Rect Setup (GL)

```c
prog = glamor_use_program_fill(pixmap, gc,
   &glamor_priv->poly_fill_rect_program,
   &glamor_facet_polyfillrect_130);

if (!prog)
   goto bail_ctx;

/* Set up the vertex buffers for the points */

v = glamor_get_vbo_space(drawable->pScreen, nrect * sizeof (xRectangle), &vbo_offset);

glEnableVertexAttribArray(GLAMOR_VERTEX_POS);
glVertexAttribDivisor(GLAMOR_VERTEX_POS, 1);
glVertexAttribPointer(GLAMOR_VERTEX_POS, 4, GL_SHORT, GL_FALSE,
                     4 * sizeof (short), vbo_offset);

memcpy(v, prect, nrect * sizeof (xRectangle));

glamor_put_vbo_space(screen);
```
Rect Drawing

glamor_pixmap_loop(pixmap_priv, box_x, box_y) {
    int nbox = RegionNumRects(gc->pCompositeClip);
    BoxPtr box = RegionRects(gc->pCompositeClip);

    glamor_set_destination_drawable(drawable, box_x, box_y, TRUE, FALSE, prog->matrix_uniform, &off_x, &off_y);

    while (nbox--) {
        glScissor(box->x1 + off_x,
                  box->y1 + off_y,
                  box->x2 - box->x1,
                  box->y2 - box->y1);
        box++;
        glDrawArraysInstanced(GL_TRIANGLE_STRIP, 0, 4, nrect);
    }
}
Glamor for Render

• Current code
  – Optimized compositing
  – Lots of CPU overhead

• Future plans
  – Ponies and rainbows
Require GL Support for Glamor

- GLSL 1.20
- Desktop GL
  - GL 2.1 or later
- GLES
  - GLES 2.0 or later
  - GL_EXT_texture_format_BGRA8888
Optional GL Support for Glamor

- GLSL 1.30
  - Integers
  - Instancing for vertex generation
- KHR_debug
- MESA_pack_invert
- EXT_framebuffer_blit
- ARB_map_buffer_range
- ARB_buffer_storage
- NV_texture_barrier
Glamor Projects

• Rework pixel format code
  – Issues with multiple PictFormats (which Gtk+ does)
  – Take advantage of texture swizzle extension
• Remove “optimization” for single-texture pixmaps
• Render text rewrite
  – Remove temporary add buffer
  – Implement new glyph cache
  – ARB_blend_func_extended for component alpha
• Fragment shader trapezoids
• Use VAOs
• Finish core context work
  – Fix render code to use VBOs/VAOs