Extension Overview

- Meets many developer requests for advanced blending features in WebGL
- Similar to "framebuffer fetch", but can be implemented on almost all devices on the internet
  - Rather than fetching the framebuffer after hardware blending operations, the application loads and stores directly from its own user-defined local storage
  - Enables fully programmable blending
- Extension is drafted and fully implemented in ANGLE
  - Expect rapid progress to completion and community approval

<table>
<thead>
<tr>
<th>GLSL</th>
<th>WebGL API</th>
</tr>
</thead>
<tbody>
<tr>
<td>#version 300 es</td>
<td>// Enable the extension.</td>
</tr>
<tr>
<td>#extension GL_ANGLE_shader_pixel_local_storage : require layout(b=0, rgba8)</td>
<td>const pls = gl.getExtension(&quot;WEBGL_shader_pixel_local_storage&quot;);</td>
</tr>
<tr>
<td>med</td>
<td></td>
</tr>
<tr>
<td>medium uniform pixelLocalANGLE framebuffer;</td>
<td>gl.bindFramebuffer(gl.FRAMEBUFFER, fbo);</td>
</tr>
<tr>
<td>medium in vec4 src;</td>
<td>pls (0/<em>binding index</em>/, texture, 0/<em>level</em>/, 0/<em>layer</em>/);</td>
</tr>
<tr>
<td>void main() {</td>
<td>// Issue a rendering pass using pixel local storage.</td>
</tr>
<tr>
<td>medium vec4 dst = pixelLocalLoadANGLE(framebuffer);</td>
<td>pls.beginPixelLocalStorageWEBGL([pls.LOAD_OP_ZERO_WEBGL]);</td>
</tr>
<tr>
<td>medium vec4 color = customBlend(src, dst);</td>
<td>glVertexArrays(...);</td>
</tr>
<tr>
<td>pixelLocalStoreANGLE(framebuffer, color);</td>
<td>glVertexArrays(...);</td>
</tr>
<tr>
<td>}</td>
<td>pls.endPixelLocalStorageWEBGL([pls.STORE_OP_STORE_WEBGL]);</td>
</tr>
</tbody>
</table>
Implementation Details

- **Supported natively on Tile Based Deferred Rendering architectures:**
  - GL_EXT_framebuffer_fetch / GL_QCOM_tiled_rendering
  - GL_EXT_shader_pixel_local_storage
  - VK_EXT_rasterization_order_attachment_access
  - Metal programmable blending

- **Fully supported on Immediate Mode Rendering architectures via shader image load/store and fragment shader synchronization:**
  - GL_ARB_fragment_shader_interlock / GL_NV_fragment_shader_interlock
  - GL_INTEL_fragment_shader_ordering
  - VK_EXT_fragment_shader_interlock
  - D3D 11.3 rasterizer order views
  - Metal raster order groups

- **“Noncoherent” mode can be supported on even more devices, but requires glPixelLocalStorageBarrierANGLE() between overlapping draws:**
  - Vulkan subpass loads (Works on 100% of webgl2 devices running the vulkan backend)
  - D3D 11 unordered access views (Works on 100% of webgl2 devices running the D3D backend)
  - GL_NV_texture_barrier
  - OpenGL ES 3.1 core, 4.2 desktop via shader images (already implemented in ANGLE)
  - Metal read/write textures
  - Metal texture barriers
Advanced Blend Demo!

https://registry.khronos.org/webgl/sdk/demos/rive/bubbles.html
Rive Rendering Demos!

- Rive’s upcoming web runtimes will be built on WEBGL_shader_pixel_local_storage
- Web apps can drive Rive animation state machines from high level JavaScript or Dart (Flutter)
How to use it

- ANGLE_shader_pixel_local_storage is checked in and enabled by default in ANGLE and Chromium
- **WebGL implementation** (WIP)
  - Landing soon!
  - Launch Chrome Canary with “--enable-webgl-draft-extensions”
- **Emscripten bindings** (WIP)
A recording of this presentation will be available at https://www.khr.io/105

For more information on WebGL, please visit https://www.khronos.org/webgl

Email: public_webgl@khronos.org