KHRONOS GROUP

WebGL Advances and Updates

Ken Russell, Google; Chris Dalton, Rive

WebGL Advances and Updates

Quarterly Khronos-hosted WebGL+WebGPU Meetups

Join WebGL+WebGPU Communities

WebGL 2.0 Available Universally

Browser Implementation Updates

New and Upcoming Extensions

Quarterly WebGL+WebGPU Meetups

Khronos hosts free, quarterly, online meetups covering the WebGL and WebGPU APIs

A great way to keep up on the advances in the web's graphics ecosystem

To receive news about these:

- **Sign up for Khronos' newsletter at https://www.khronos.org/events/**
- **Follow https://twitter.com/WebGL**

Note: no WebGL BOF at SIGGRAPH this year. Next meetup is October 4.

Join WebGL+WebGPU Communities

The WebGL and WebGPU APIs are supported by vibrant online communities!

If you're developing with these APIs, we would like to hear from you!

WebGL:

- Join the WebGL Dev List: product announcements, demos, new tools, job postings, questions, discussions - all are welcome!
- Khronos' public_webgl mailing list hosts lower-traffic spec announcements
- Talk with browser implementers and other developers in the WebGL Matrix chat room

WebGPU:

- The WebGPU repository describes how to provide API feedback to the community group
- Talk with browser implementers and other developers in the WebGPU Matrix chat room

A lot of cool stuff shows up on #webgl and #webgpu on Twitter 😎

WebGL 2.0 Available Universally

Safari 15 delivered WebGL 2.0 on iOS and macOS in Fall 2021! 🎉

WebGL 2.0 is now considered universally available

As an application author, you can now target WebGL 2.0 with confidence

WebGL 2.0 is a robust API target - fixes many corner cases compared to WebGL 1.0 + extensions

We encourage you to migrate to WebGL 2.0

It's no longer necessary to maintain a WebGL 1.0 fallback path unless you need to reach absolutely every device

- In particular, older Windows machines and Android devices

Browser Implementation Updates

Chrome and Safari engineers are actively collaborating on the WebGL implementation on top of Metal, used on both macOS and iOS

Chrome and Firefox engineers are implementing display-p3 support for WebGL-rendered canvases, and looking toward HDR support

WebGL CTS improvements and associated browser bug fixes are resulting in improved portability for applications

New and improved extensions are adding functionality and increasing performance

Upcoming WebGL Extensions

GL_ANGLE_shader_pixel_local_storage(_coherent)

- **Motivated by many developer requests for more 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 implemented with full coherency in the ANGLE GLSL compiler**
  - Expect rapid progress to completion and community approval

GLSL

```
#version 300 es
#extension GL_ANGLE_shader_pixel_local_storage : require
layout(binding=0, rgba8)
mediump uniform pixelLocalANGLE framebuffer;
mediump in vec4 src;
void main()
{
    medium vec4 dst = pixelLocalLoadANGLE(framebuffer);
    medium vec4 color = customBlend(src, dst);
    pixelLocalStoreANGLE(framebuffer, color);
}
```

OpenGL ES API

```
// Set up pixel local storage.
glBindFramebuffer(GL_FRAMEBUFFER, fbo);
glFramebufferTexturePixelLocalStorageANGLE(0/*binding index*/,
                                           texture,
                                           0/*level*/,
                                           0/*layer*/);

// You could also do memoryless storage!
glFramebufferPixelLocalStorageANGLE(1/*binding index*/,
                                    GL_R32F);

// Issue a rendering pass using pixel local storage.
glBeginPixelLocalStorageANGLE(1, GLenumArray({GL_ZERO}));
glDrawArrays(...);
glDrawArrays(...);
glEndPixelLocalStorageANGLE();
```

Upcoming WebGL Extensions

GL_ANGLE_shader_pixel_local_storage(_coherent) (continued)

- **Supported natively on Tile Based Deferred Rendering architectures:**
  - GL_EXT_framebuffer_fetch / GL_QCOM_tiled_rendering
  - GL_EXT_shader_pixel_local_storage
  - VK_ARM_rasterization_order_attachment_access
  - Metal 2 Raster Order Groups
- **Fully supported on Immediate Mode Renderering 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
- **"Noncoherent" mode can be supported on even more devices, but requires `glPixelLocalStorageBarrierANGLE()` between overlapping draws:**
  - Vulkan/Metal core via attachment access
  - D3D 11 core via unordered access views
  - GL_NV_texture_barrier
  - OpenGL ES 3.1 core, 4.2 desktop via shader images (already implemented in ANGLE)

Upcoming WebGL Extensions

EXT_provoking_vertex

- When using the `flat` interpolation qualifier in shaders, it's necessary to precisely define the initiating, or provoking, vertex for a given primitive like a triangle or line
- Unextended OpenGL ES 3.0 / WebGL 2.0 require OpenGL's convention of the "last" vertex as the provoking vertex
- Emulating this is expensive on some platforms - Direct3D and Metal especially
- This extension allows the provoking vertex to be configured by the application, specifically to the first vertex
- Significantly improves performance of flat shaded geometry on some platforms
- Extension is in draft form now; expect rapid progress to community approval

Upcoming WebGL Extensions

OES_draw_buffers_indexed

- **Enhances multiple draw buffer functionality**
- **This extension provides the ability to:**
  - enable or disable blending
  - set the blend equations
  - set the blend functions
  - set the color write masks
  - all per color output!
- **This extension was specifically requested by the 3D Formats working group to implement advanced materials (e.g., that use dual depth peeling) more efficiently**
- **Community approved; in some browsers already, coming to all soon**

Upcoming WebGL Extensions

Base Vertex/Base Instance & Multi-Draw Variation

- Provide control of BaseVertex, for indexed draw calls, and BaseInstance, for instanced draw calls
- Multi-draw variants are provided as well
- Allow reuse of index buffers to draw multiple different geometries from the same set of vertex buffers
- Can reduce CPU overhead and memory consumption, and improve performance
- If you've needed these draw parameters, please try the extensions and provide your feedback
- Can be tested in Chrome today by enabling WebGL Draft Extensions in `about:flags`
  - Please file any bugs on crbug.com
- Will come to all browsers shortly after community approval

Q&A

Q&A will follow the rest of the Fast Forward presentations!