WebGL and WebGPU Updates

Ken Russell, Google and Kelsey Gilbert, Mozilla
On Behalf of the WebGL WG and WebGPU CG
WebGL+WebGPU Meetup, April 26, 2022

Agenda

Join WebGL & WebGPU Communities

WebGL Updates

- Firefox WebGL Updates
- Chromium WebGL Updates
- WebGL 2.0 in Safari / Chrome Updates
- Recent ANGLE/Metal Improvements
- New & Upcoming WebGL Extensions

WebGPU Updates

- Standardization
- Implementations
- Resources and Contributing

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!
- On the WebGL side:
  - Please consider joining the WebGL Dev List: announcements of products, demos, new tools, job postings, questions, discussions - all are welcome!
  - Khronos' public_webgl mailing list hosts lower-traffic spec announcements
  - The WebGL Matrix chatroom offers a way to talk with browser implementers and other developers
  - You can find a lot of cool stuff by searching #webgl on Twitter 😎

Join WebGL & WebGPU Communities

- On the WebGPU side:
  - If you have feedback on the API, please see the main WebGPU repository for options to communicate it to the community group
  - The WebGPU Matrix chatroom also offers a great way to talk directly with browser implementers and other developers
  - There's an increasing amount of cool stuff showing up on #webgpu on Twitter 😎
- We all look forward to hearing from you!

Firefox WebGL Updates

- Implementing the Canvas Color Space Proposal
- Internal builds have WebGL-rendered P3 canvases displaying to the screen on macOS!
- Fixes for several fuzzing and/or security bugs
- Performance improvements, especially for video->texture uploads
- Iterating on implementations of extension proposals
- OffscreenCanvas is in the process of rolling out!

Chromium WebGL Updates

- Chris Cameron is also implementing the Canvas Color Space Proposal
- Lots of updates to Chromium's texture uploading code
- Making good progress; will soon have the flexibility to support nearly any desired color space (including the specified ones, P3 especially)

WebGL 2.0 In Safari / Chrome Updates

- Safari and Chrome share ANGLE's Metal backend as the implementation for WebGL 1.0 and 2.0
- Safari is already shipping this on macOS and iOS
- Chrome is still using OpenGL, but in the process of switching to Metal

Recent ANGLE/Metal Improvements

- Multisample resolution bugs discovered by Unity, Three.js, Gregg Tavares
- Major performance improvements to glReadPixels performance
- Memory usage improvements during canvas resizing
- Vertex data handling fixes
- Support for BaseVertex/BaseInstance extensions
- Fixes for various object leaks
- Improvements to dual-GPU support
- Performance regressions using large buffers
- Fixes to occlusion queries
- These and others are coming to all browsers, especially iOS, as soon as possible
- Joint work (thank you!) with:
  - Alexey Knyazev (Independent), Geoff Lang (Google), Gregg Tavares (Google), John Cunningham (Apple), Jonah Ryan-Davis (Google), Kimmo Kinnunen (Apple), Kyle Piddington (Apple)

WebGL 2.0 In Safari / Chrome Updates

- You can test Chrome on top of ANGLE's Metal backend today:
  - Go to `about:flags`
  - Set "Choose ANGLE graphics backend" to "Metal"
- (please test it in Chrome Canary - fixes are actively going in)
- and compare its behavior to the OpenGL backend
- Some regressions exist, like low-power/high-performance GPU selection not working yet
- Please file any bugs you find with the Metal backend on ANGLE's issue tracker
- Please test Safari Technology Preview and file any bugs you see in WebGL on bugs.webkit.org, component "WebGL"
- (For other browsers' bugs, consult "How to get a WebGL Implementation")

WebGL 2.0 In Safari / Chrome Updates

- WebGL 2.0 can now be considered universally available across browsers, operating systems and devices
- As an application author, you can target WebGL 2.0 with confidence
- WebGL 2.0 has resolved many corner cases and behavioral differences compared to the combination of WebGL 1.0 + its many 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

New 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
- Shipping in Chrome; coming to other browsers
  - Please file any bugs on crbug.com, WebGL component

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
- Reduce CPU and memory overhead in certain scenarios
  - Have customer feedback that these can improve performance as well
- 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

WebGPU

- An upcoming “modern” style graphics API for the Web
  - “Prevalidated” style - pipeline objects, bind groups
  - Compute shaders, shader storage
  - No global state
  - … and much more
  - Foundation for future features like bindless, raytracing, shader features

- Under development on GitHub and at the W3C
  - Thank you to Khronos for hosting us here!

WebGPU - Standardization Updates

- Standardization continues; conformance testing in high gear
  - Aiming to reach 1.0 around Q3 2022 (spec and conformance tests)
- Shading language under rapid development
  - Feature completeness and numerous language refinements
  - Ergonomics (reduced verbosity, type inference, compile-time expressions)
  - Static analysis of control flow and uniformity
  - float16 extension!
  - https://www.w3.org/TR/WGSL/
- API spec driving toward 1.0 - recent areas of focus:
  - Video and canvas interop, color management
  - Privacy and security
  - Ergonomics, lifting restrictions, defining optional features
  - Bringing the spec up to date, fleshing out details
  - https://www.w3.org/TR/webgpu/

WebGPU - Implementation Updates

- Available to try today in Chrome+Firefox
  - For local development, test the latest browser code:
    - Chrome Canary: **`enable-unsafe-webgpu`** in **`about:flags`**
      Firefox Nightly: set **`dom.webgpu.enabled`** in **`about:config`**
      (don't leave these enabled while browsing the web)
  - Mostly, but not fully, interoperable, due to changing spec
- Chrome
  - Origin Trial allows you to publish WebGPU apps directly to end users
    - Mac, Windows, Chrome OS
    - Extended already through at least Chrome 105 (Sep 21, 2022)
    - **Breaking changes by design - you must keep your content up to date**
    - On Chrome Stable, so you may need temporary polyfills for newer API changes
    - Instructions: web.dev/gpu
  - Aiming for 1.0 release later this year
    - Linux/Android soon afterward

WebGPU - Resources

Articles:

- web.dev/gpu Tutorial on getting started with WebGPU
  - More articles linked from here

Samples:

- Austin Eng's WebGPU Samples (Github)
  - Journeys from your first triangle, to real-world compute & graphics examples
- Brandon Jones' Clustered Shading, Metaballs, and Spookyball demos
  - Real-world usage of the WebGPU API, compute shaders, and rendering techniques
- Shrek Shao's WebGPU Deferred Renderer
  - Shows how to do deferred shading in WebGPU, complete with debug views

WebGPU - Resources

Projects with WebGPU backends well under development:

- Babylon.js, Three.js, TensorFlow.js, and others
- wgpu and the ecosystem of Rust WebGPU projects

Shader compilers

- Compile from { WGSL, Vulkan SPIR-V } to { WGSL, SPIR-V, HLSL, MSL }
- Good for:
  - Seeing what the WGSL language looks like
  - Converting your existing shaders
    - Use Glslang to compile existing GLSL to Vulkan SPIR-V
      - (glslangValidator or glslc from the Vulkan SDK)
- Google's Tint: https://dawn.googlesource.com/tint
- Mozilla's Naga: https://github.com/gfx-rs/naga
- Preview them on Shader Playground: https://shader-playground.timjones.io/

WebGPU - Contributing

Contributions welcome!

- Try the API! File API issues and browser bugs, and let us know what you think!
- Try out WebGPU via Babylon.js, Three.js, TensorFlow.js, etc.
- Publish previews with Chrome's WebGPU Origin Trial
- Help with conformance testing
- Contribute samples/demos using WebGPU
- Join the conversations on the Matrix chat!