Shipping WebGPU on Android
March 20, 2024

Brandon Jones, Google
WebGPU is shipping on Android!

Android in Chrome 121 (January 2024!)

- Initially on Android 12+
- ARM and Qualcomm GPUs
- More OS versions, GPUs planned.

Shipped on Windows/Mac/ChromeOS in Chrome 113 (April 2023)
Demo!

https://playcanvas.com/demos/arealights/
Dawn's Vulkan backend

Vulkan is used to power WebGPU on Android and ChromeOS. Eventually will ship Linux with a Vulkan backend too.

Baseline requirements for WebGPU support:

- Vulkan 1.1
  - *Or* Vulkan 1.0 w/ VK_KHR_maintenance1 & VK_KHR_maintenance2
- fragmentStoresAndAtomics & fullDrawIndexUint32 features
- maxImageArrayLayers limit ≥ 256
- BC *or* (ETC2 && ASTC) texture compression
Technically works on Pixel 1! (Not shipping)
Vulkan Highlights!

From Corentin Wallez and David Neto:

- Vulkan spec is VERY detailed and precise
- Validation layers work
- Vulkan group @ Khronos is very responsive to questions and bugs
Vulkan Challenges

WebGPU is an API, not a game, app, or engine

Need to ensure that every possible code path is reliable

- Don’t have the luxury of skipping an effect on a buggy piece of hardware
- We see ALL the driver bugs

Vulkan is a Bring-Your-Own-Utils API, so Dawn got to build all the same helper classes as everyone else
Vulkan

OpenGL 3+

OpenGL 1.x

DX9

DX11

DX12

"Hello Triangle":
- GL1 uses SDL
- GL3+ and Vulkan use GLFW3.

Admittedly the Vulkan version has more comments and the GL versions could do more error checking.
Vulkan is our biggest backend

Metal: 6k
D3D12: 9k
Vulkan: 13k LOC

+Lots of common code in our platform-agnostic frontend.
Shipping on Android
Highlight: It mostly worked first try!

Majority of Dawn’s Vulkan backend simply worked when we got it running on Android.

Testament to the portability of Vulkan and the work of our team.

Majority of issues were around resource sharing with Chrome/between processes.

Plenty of new and exciting driver bugs too.
Robust WebGPU CTS was invaluable!

112k+ conformance test cases

Helps identify both Dawn implementation issues and driver issues

Lesson learned from WebGL, which has its own extremely valuable CTS test suite.
Canvas format issues

WebGPU spec guarantees canvases can allocate rgba8-unorm and bgra8-unorm surfaces.

Due to Chrome’s multiprocess architecture, any canvas rendering must go through an AHardwareBuffer.

AHardwareBuffer has an RGBA8 format, but no BGRA8 format.

Solution was to do a just-in-time copy to the AHardwareBuffer-backed textures if needed.

Devs can avoid overhead by checking the navigator.gpu.preferredCanvasFormat()
Splitting command buffers for fun and profit

Frequent crash on one vendor when a texture modified in a compute shader was sampled or written to in a subsequent render pass.

But ONLY if they were in the same VkCommandBuffer.

Fortunately, Dawn doesn't build VkCommandBuffers directly from user calls.

- Calls are recorded into intermediate format and replayed into VkCommandBuffers at submit time.
- Allows us to detect the issue when building command buffers and silently split them.
SPIR-V fixes when cross compiling from WGSL

Several issues were able to be resolved completely within our WGSL cross-compiler, Tint

Passing Matrices as function args was causing a crash on one vendor’s drivers

Re-written by Tint to be a pointer

Would have been much more difficult to work around if we consumed SPIR-V directly
Sometimes workarounds are impractical

On some devices passing an index of 0xFFFFFFFF (primitive restart value) with a “-list” topology would trigger a device loss.

No reasonable way to detect it. Definitely not going to scan every index buffer before every draw, and the device loss triggers before the associated vertex shader is called.

This is clearly bad app behavior that’s trivial developers to fix.

Device loss is annoying, but it’s not dangerous.

So... let it be! Better than blocklisting.
Enumerating the devices should be safe... right?

We know of at least one (decade old) GPU where calling vkEnumeratePhysicalDevices causes a crash.

We use information gathered from the enumerated devices to inform our blocklist.

So we need a blocklist to block some devices from checking our blocklist...
We still have work to do!

Extending Android OS and GPU support

Investigate mobile-specific optimizations

- Have exposed mobile-friendly features, like float16 shader operations.

Ongoing maintenance
Thank you!

Brandon Jones

Email: bajones@google.com

Social media, WebGPU articles, and more: https://toji.dev