Attendance from WebGPU CG:

- Google
    - Austin Eng
    - Corentin Wallez
    - David Neto
    - Kai Ninomiya
    - Ken Russell
    - Ryan Harrisson
- Mozilla
    - Dzmitry Malyshau
    - Jeff Gilbert

Notes from Shannon Woods (Google) below:

WebGPU - Corentin Wallez (Google, WebGPU chair)

Slides

WebGPU basically explicit API for the web. Has compute.
W3C-based group, active participation from all the browser vendors + Intel and others.
Design constraints
Security, portability (truly write-once, run-everywhere)
Impl. on top of D3D12, Metal, Vulkan
Remotability
High performance

Some differences from Vulkan— makes it almost like a “vulkan-lite”?
No consensus yet on shading language.
Implementations already WIP in three major browsers.
Corentin illustrates the high-level architecture of the implementations w/ some diagrams (in slides)
wgpu-native demo very-very quickly - not running in a browser, but using the native application targeting webGPU. Having a native implementation allows native development easily portable to web via wasm.

For discussion w/ group:

- Want a roadmap of features for what’s going to be required in N years for Vulkan 1.x (moving the minimum bar for Vulkan)
    - WebGPU1 has to define its own feature set by using gpuinfo.org as a crystal ball

    - Would be really nice to have a Vulkan-defined feature set instead, for a future version of WebGPU
- WebGPU uses more of the API/a bigger spread, so it's been finding corner case & filing bugs. Is there a better process?
- Would really like to map buffers cross process without loading userspace driver. (Want zero-copy uploads)
    - Tobias: Should be able to use a file descriptor?
    - Corentin/Jesse H/Jason: trouble with opaque types, DMAbuf, stuff
- Zeroing of compute shader shared memory (for security purposes). WebGPU can do it themselves (slowly), but would like an extension.
- IHV guidelines on pipeline barrier type bundling. How to batch so they are performant. Looking at source where available, but that doesn't always work for other IHVs.
- Discovering support for format capabilities across hardware.
    - Tobias: Sascha Willems is working on that as we speak.

Pipeline barrier type bundling

Tobias: We tell developers to batch everything, as best they can. Developers haven't asked that, so surprised to hear it coming from WebGPU. Just do your best, doesn't matter. If you can batch, then do.
Corentin: We have choices to make— we're splitting command buffers; do we wait for more to batch them, or send as we have them?
Tom: Some of these issues are going to need time and clarification— can people with access file an issue? Could create a public issue in Vulkan-Docs repo, and we'll work it there?

Public Roadmap

Tom: Public roadmap is a strong term— but we've had some discussions since the last F2F on the topic of moving the minimum bar.
Tobias: Right, but we don't expect any seismic shifts there.
Corentin: Would be good to be able to say— if there's a feature being added to WebGPU, if we know that in 5 years or whatever, it will be required in Vulkan
Tobias: That's not the model that Vulkan uses— we're not going to require hardware features
Tom: Tobias's statement might be a little bit harder-line than we've discussed, but that's the general idea.
Tobias: What we do in this group is facilitate discussion between vendors on market pressures & feature additions, but we don't force hardware advances.

[Shannon's aside: "feature soup" gets said a lot, and seems to have established itself as a working group term. Heh.]

Tobias: Part of the reason I'm resistant to defining this inside of the API is — if you define it within the API, the adoption goes down. If you stick too many things together, no one can support it. The benefit of "feature soup" is that everyone can support something. If you can figure out what the adoption of something is, it doesn't matter whether Vulkan has required it, as long as it's widespread enough.

Shannon W: Is WebGPU asking for the Vulkan WG to *cause* hardware features to ship, or to forecast what we believe will get shipped anyway?
Corentin: The latter, not the former.
Tobias: Ah, ok— it's the forcing function that I object to. There's stuff like this we're working on I can't talk about, but we would like to help that happen.
Daniel K: Right— but even if we have such a roadmap, we can't make it public, right?
Tobias, Tom: Well, maybe a bit more public, if not specific.
Alon: When OpenGL ES was driving features, we had quorum about this, but because we're in a single more diverse group now, we've left the platform owners to do this definition. Each vendor has their own definition, it seems like.
Tobias: Right, GLES was kind of a proxy for Android, and GL for desktop. But now you probably need to talk directly to Android, or talk directly to DirectX.
Jason E (Intel): When we have been able to require something across existing hardware, we have. But when you talk about forecasts— that's REALLY hard to do, because there's so many different parties involved.

Shading Languages

Tom: What's WSL?
Corentin: Web Shading Language — A high-level shading language made by Apple for WebGPU.
<room in general grimaces>
Tom: Where does the translation happen? Does the webpage have to do the translation itself?
<Corentin explains while Shannon gets behind on notetaking>
Corentin: To be clear, we will never ask drivers to implement support for WSL
Tobias: Separating high-level language from SPIRV was one of the best things we did for Vulkan. Ingesting a high-level language at whatever intake point opens you to parser bugs, etc.
Corentin: Most impactful feedback would be from ISVs.
Jesse B (Unity): We care about HLSL
Eric B (Adobe): Creating a new high level language is a cardinal sin. Don't. Do. That. Don't want to rewrite all my shaders AGAIN.
Jesse B: If we can transcode to HLSL to whatever you need, great. If we can't, we may not support your platform at all.
Eric B: Would really not like even to write another transcoder. If there's an existing tool to get to an intermediate representation, that's good. Would suggest SPIRV is an EXCELLENT existing intermediate representation.

Dmitry: Sounds like ISVs would accept Web-HLSL because HLSL is cool?
Matthaeus: HLSL doesn't have a spec, so how do you know WHLSL is even close?
Jesse B: Most Unity devs aren't even writing shaders by hand— they have a shader editing tool.
Neil T: WebGL already has GLSL shaders— need to be able to transcode that to your representation, or you can't show old content.
Tom O: We got so much leverage out of SPIRV— there are people who desperately want HLSL, and there are people who desperately want something else. Not good to have wrong level of abstraction. Highly recommend SPIRV.
<some discussion of fuzzing, tools, optimizers & who maintains it>
Corentin: Our implementation will use spirv-opt, spirv-val; other implementations may choose differently.
<person I don't recognize>: Doesn't code written by Khronos then become part of an attack surface on the web?
Alan: We're aware of this— this is part of the reason we're fuzzing so heavily. We think everyone benefits from this.
Jason: One comment on fuzzing & validation strategy — if we're using the same validator on the web, we need to make sure that it's very conservative WRT spec issues. Don't want to get into a position where something is passed as valid by validator & fuzzer but crashes a driver, then someone opens a spec issue which gets ignored for six months while people's code is crashing drivers.
Corentin: We're aware we need to be attentive, that there will be driver bugs, and we'll need to disable stuff.
Alan: David added robust buffer access to do address clamping; we have a lot of experience built up, so people don't have to start w/ a clean slate. Ppl can take the lessons we've learned over time and apply them here.
Tom's summation: IHVs are nervous about the prospect of hostile SPIR-V. Not sure what to do about it.
Corentin: You don't need to do anything— that's the browser's job. But SPIRV-fuzz will help.