GPU Web 2019-07-08

Chair: Corentin & Dean
Scribe: Ken, Austin
Location: Google Meet

Tentative agenda

- Draft WG Charter
- PR Burndown
- #301 Add GPURenderBundle
- Bind Point Renaming API proposal 351
- gpuweb hosting webgpu.h headers
- Agenda for next meeting

TL;DR

- Please show the draft working group charter to legal if you need to!
- GPURenderBundle PR accepts as is, do follow-up PRs to discuss extending the set of commands they support.
- Bind Point Renaming API proposal #351
    - This adds an adapter to convert from HLSL to WebGPU binding models but also allows for "WebGPU first" WHLSL to work without the adapter.
    - Microsoft explained that the "spaceN" attribute was added to HLSL to represent multiple unbounded arrays of descriptors.
    - Concern that the semantic of spaceN in HLSL doesn't match the proposed WebGPU semantic.
- Hosting webgpu.h in the gpuweb org.
    - Mozilla and Google want to have interoperable implementations to help applications port to WebGPU in WASM would like to GPUWeb to host the header.
    - Concern that this isn't in the scope of the group that spec only a web API. Decision to not host the header there.
- PR #292 approved, #318 closed. Bitmasks don't have a suffix.

Attendance

- Apple
    - Dean Jackson
    - Justin Fan
    - Myles C. Maxfield

    - Robin Morisset
- Google
    - Austin Eng
    - Corentin Wallez
    - Idan Raiter
    - Ken Russell
    - Shrek Shao
    - Ryan Harrisson
- Intel
    - Yunchao He
- Microsoft
    - Damyan Pepper
- Mozilla
    - Dzmitry Malyshau
- Elviss Strazdiņš
- Mehmet Oguz Derin
- Timo de Kort

Draft Working Group Charter

- https://htmlpreview.github.io/?https://github.com/grorg/admin/blob/wg-charter-draft/wg-charter.html
- Pull request https://github.com/gpuweb/admin/pull/15
- DJ: sent to Apple Legal for review. Please comment on the pull request. Also has to be sent to the W3C.
- CW: So all of us will have to go through legal to join the WG?
- DJ: will propose to W3C advisory committee that we want an official WG, and here's the charter. Thought it was a good idea to get Apple Legal review before sending to the W3C. Would be good to get pre-review by others' legal teams.

Spec Editors

- DJ: CW and I will have a talk about it. We think 4 editors would probably be too many.
- JG: does seem like a lot.
- DM: how many does WebGL have?
- JG: 2.
- CW: 3 could maybe work but it dilutes the responsibility. Need to talk and think more. In W3C land the chairs have to decide the spec editors. Feels awkward because we don't want this to be a top-down decision. Will discuss proposal for how we decide. Spec editors are supposed to just implement decisions for the group. Will discuss and come back to the CG.

#301 Add GPURenderBundle

- CW: think we'd mostly agreed this was ready to land. Any other concerns?
- MM: the conversation's been spread out for many weeks. Where did we end up?
- CW: we add RenderBundle concept. At first, add only commands in D3D12 bundles, Metal's Indirect command buffers, and Vulkan's secondary command buffers. Start with the intersection of all of these, and figure out later how to extend the feature set. The RenderBundles being the intersection of all of these APIs seem to have agreement to move forward.
- CW: RC's not here but we had his agreement on the PR both online and offline.
- DM: the plan to land it as-is and then expanding capability later seems reasonable. Rafael did have some concerns but he did approve the PR.
- DP: MSFT's point is that we don't expect any impls to use D3D12 bundles directly. Could use command lists, etc. So given that, we could definitely grow and expand functionality later.
- MM: we would like to use Metal's indirect cmd buffers. Not sure it makes sense to expand it. If we want a HW-accelerated bundle that's great, but the moment we add something that can't be handled in hardware we fall back to software and have diff perf characteristics.
- CW: currently it's a strict subset of the Metal indirect command buffers.
- MM: guess I should save my comments for the point where someone tries to add something to this.
- CW: let's discuss when we have more commands to add to the render bundle.
- DM: #286 contains comparison of what the different APIs support. It's currently an intersection of this.
- MM: FF is the only browser that will run on D3D that isn't Chrome based and has WebGPU support. Are you OK with using command lists rather than bundles?
- DM: I want to have support for both and see if it makes a difference. Currently we have the luxury of making this comparison.
- CW: later we could discuss possibility of expanding to full Metal indirect command buffers.
- CW: need to have future discussions about adding stuff to render bundle.
- CW: will look to see if needs any updating for recent changes and land this tomorrow.

Bind Point Renaming API proposal 351

- CW: thought we already discussed this last week?
- MM: yes, was mostly an introduction.
- MM: one piece missing: a little context: (recap from many months ago) there is a difference between HLSL's binding model and WebGPU's binding model today. At least 3 ways to solve it. 1) Change WebGPU's binding model to match HLSL; 2) change HLSL's to match WebGPU's; 3) add an adapter. My original PR was to change

WebGPU's binding model to match HLSL's, but after long discussion it was closed, mainly because: in HLSL source it's common for all resources to be in the same space, and we don't want to put those all in the same WebGPU bind group.

- MM: changing HLSL's binding model: want to be source compatible with all HLSL out there.
- MM: so we have the adapter proposal.
- CW: originally the PR was a dictionary, HLSL register names as strings mapping to WebGPU binding sets. Now it's less strongly typed. Damian you had concerns with this?
- DP: discussions were around, where should adapter even live. I'm thinking that the adapter should live outside WebGPU, and the shading language should use WebGPU's binding model concept. If you want to use existing D3D11 HLSL shaders you have to run them through some offline shader translation step.
- JG: modern HLSL has this, right? This is a problem of taking D3D11 era shaders and run those in WebGPU? Isn't the problem that it's not inefficient, but there are namespace collisions?
- DP: what is "this" in context?
- CW: WebGPU has a 2-dimensional binding models. Descriptor tables & offsets into them. HLSL has a 3-dimensional binding model. Register name, space, and register type (s/t/u/c). If you remove the third dimension of type, then it's 2-D. The register number is an offset into a descriptor table. Index is offset of descriptor table in root signature.
- DP: … corresponds to bind groups?
- CW: yes. Similar to how Chrome and Firefox translate bind groups into descriptor tables. (...more…)
- DP: you're suggesting space is used to pick descriptor table?
- CW: yes.
- DP: what's story for people writing shaders to WebGPU? "space" is translated into bind group?
- CW: WebGPU hasn't settled on shading language yet. On SPIR-V side things are easy because it has the facilities to work with Vulkan and it's similar to WebGPU. On WHLSL side, the binding model would be mapped with the "space" declaration.
- DP: the PR here is assuming the source language is HLSL, right? If WebGPU SL is not WHLSL then we wouldn't accept this PR, correct?
- CW: correct.
- JG: we wouldn't accept it today.
- MM: right now SPIR-V's binding model matches WebGPU's.
- JG: you could do this today. Main problem of mapping HLSL's normal declarations to WebGPU's is that you could have something of a different type in the bind group coordinate.
- MM: it's really common. b0 and t0 collide.
- JG: easy for HLSL shaders to work around this issue. Could forbid this sort of common thing from HLSL.

- DP: you could do that. Solves half the problem. The other thing you expect engine developers to do is split shader variables into bind groups. These ones all together, those all together, etc. Without that extra bind group parameter you don't have that.
- JG: that's not my understanding. You have that in HLSL via the space?
- CW: that's what the PR does. It doesn't have spec text, but it says if a register isn't in the translation table then it's used as is with the space declaration (...) and assuming no collisions. Very much like the root table except that it's a 2D thing.
- DP: worried that this will lead to wrong usage. Main use of "space" is for unbounded arrays. In D3D12 you can have an unbounded array. As large as the descriptor heap. Can use dynamic indexing to pick any entry based off an initial register. If you want multiple unbounded arrays (one of tex2Ds, one of tex3Ds, etc.) you'd use "space" to disambiguate. Seen titles that have 65K tables and they alias them. We should be careful about saying a space "is" a bind group because they're not the same thing.
- MM: thanks for this input. Would like to flip around: say you (DP) were designing HLSL interop. How should it match?
- DP: WHLSL should use terms WebGPU uses. Should talk about bind points and bind groups. We should have a converter which can convert HLSL to WHLSL using generic rules. dxc has this. We should provide a cheap way for people to do this online in the browser, but the expectation is that people should do that offline.
- MM: the counterpoint is that existing shaders don't work out of the box.
- DP: true. Shader snippets will work out of the box, just have to write the binding stuff.
- JG: that's fair. Want to think more about the differences between what these things mean.
- CW: thanks for your input. So, we should use the same terms D3D12 HLSL uses. Would confuse everyone if we used "space" to denote a bind group.
- DP: yes. It's more than a naming thing. Myles' PR has this. You have to say what bind group each register goes to.
- MM: this is a tradeoff. Most shaders in our corpus don't specify space at all. Have 7 resources, going to use all 7. No unbounded arrays, so works with WebGPU. Right now WebGPU has no unbounded arrays because each bind group has to have a determined set of resources in it. Not sure the benefit of this offline compilation is worth the developer story of having to run this extra compilation step.
- KN: it's only a compilation step for existing D3D11 shaders. Don't want to hyperoptimize for the case where they are trying to run existing shaders unmodified rather than writing the ~12 lines of binding code at the top of the shaders.
- MM: it's a question of degrees. HLSL is complex enough that we won't get 100% compatibility. How many paper cuts will developers need to have?
- KN: not really a paper cut. Your shader might have many potential incompatibilities with WHLSL.
- DP: think there's value in enumerating compatibility we want. Someone might have a D3D11, or a D3D12 shader. Shader they're compiling to SPIR-V. All of these have different tradeoffs, worth considering.
- MM: how many developers are in the last camp?

- DP: not sure. Google contributed HLSL to SPIR-V.
- CW: Google might have a better idea of how many developers are using that path. DN might have a sense; he's out this week.
- CW: seems we need more data, how many developers are in each bucket. Also need to think about what Damyan explained.
- MM: sounded like you'd like an offline compiler to rewrite register keywords to not collide. What characteristics would the result have? Different syntax? Same syntax but meaning of "space" is something else?
- DP: would tune syntax to meet WebGPU's terminology. Name bind groups, etc.
- CW: what Kai said, they'd have to rewrite their binding code, but only that.
- DP: people have their shaders and only they know how they want them mapped into bind groups.
- MM: right, and this PR is a way to express that.
- CW: do you know how DXC to SPIR-V (spiregg) handles unbounded arrays?
- DP: I do not.
- DM: we don't have any investigation about unbounded arrays.
- CW: Vulkan (...) so can't have it in WebGPU.
- CW: OK if we put this on the agenda for next week?
- MM: what would we have to contribute then that we don't have now?
- CW: Google can contribute ballpark number of people using spiregg.
- JG: I'd like more time to think about how this maps to the different APIs.
- Resolved: discuss more next week.

gpuweb hosting webgpu.h headers

- CW: both Moz and Goog have WebGPU in native. There's a number of random people on the Internet that want to use WebGPU native. Partly because they want to Emscripten it. Interested in making a shared header between these impls. Where should it live? Obvious space is this group.
- JG: obvious space is Emscripten workspace.
- MM: we've already had this discussion. Why are we having it again?
- CW: because we need a place to put it.
- JG: way that people interface with WebGL is via the GL API and that lives in Emscripten. If the goal is to have a common interface then it should be there.
- KR: That's not quite accurate. Emscripten exposes the OpenGL APIs which are hosted by Khronos. Emscripten has an implementation of that API it provides to Emscripten-compiled apps with JS magic. It doesn't seem appropriate to codify shared API description in Emscripten. People are trying to compile to native library for compatibility. Emscripten would rather import this from somewhere externally.
- KN: one argument against putting it in Emscripten is that we want to import it.
- JG: we're not spec'ing the native API. Not the job of this CG. Would prefer if it happened elsewhere.

- MM: my thought exactly. The moment we include a native C header in this WG is where we start discussing native feature requests.
- DJ: we did explicitly put in our charter that we wouldn't do native APIs. I suggest making a Github repo for it somewhere that Moz and Dawn share.
- DM: don't you find it there is an analogy with WHLSL? WHLSL is not quite a product of the WG (because we haven't settled on the shading language). We're talking about something that a few members find useful.
- JG: I don't want to discuss it if it's not in the charter.
- CW: when WASM has host bindings then what will we do?
- JG: host bindings are something that happens in the glue layer. Something that doesn't concern us?
- CW: don't think we care about this group forgoing rights to this header. If it becomes a defacto standard then there won't be an opportunity for this group to change it.
- MM: comparison with WHLSL is different. The group agreed WHLSL is going to be the standardized shading language.
- JG: I understand the desire to talk about a native API, but this group decided to not talk about that. Is there a benefit to tighter integration with WASM host bindings?
- KR: It's still an amorphous concept. People on wasm and host bindings are still talking about WebIDL descriptors that aren't really the same as JS ones. Need some mapping somewhere to expose JS APIs to C. If Moz, Google choose one path and this group has no input, then when those things are standardized and they don't match, there's no chance to change it later.
- JG: I don't think the IDL thing is strictly a C header.
- KR: Presumably, but the C header would be autogenerated from the WebIDL.
- JG: Would be happy for native apps using WebGPU to be shareholders to WebGPU, but I don't think they should influence how we present the API.
- CW: …if we do it in this group, it's under the W3C CLA. If it's outside it'll be Apache2/BSD and this group will have no involvement.
- JG: Can be within; just not this working group.
- DJ: Agreed. It could even use the same CLA and webgpu.io link to it. This group is just forbidden from discussing native APIs so it's important the code live outside. I would suggest a separate Github repository.
- CW: thanks for the input. We'll figure out a different path than putting it in this group.
- DJ: I'm not saying this isn't imp't. It's a really useful topic. Thank you for doing it. I'm not arguing against it, just against it being in this group.
- CW: Yes, understand there's other constraints. DM and I will figure this out offline.
- DM: Dean has a strong point - is this reflected in the WG charter that was just published?
- CW: WG/ CG charter says this group is not meant to specify hardware algorithms and not meant to be implemented directly on hardware / GPU drivers.
- DM: that's not what Dean was saying. Dean was saying this group is forbidden to discuss native APIs.
- DJ: Said that incorrectly; What CW said.

PR Burndown

- CW: #318 have we discussed bitmasks?
    - JG: I put up a new PR. It's different now. Also have to provide a typedef. The typedef was previously "Flags". Now can't be because it collides with interface. Users will never see it.
    - CW: sounds good on our side.
    - DM: you want to proceed with #318 instead of #292?
    - JG: right. #318 makes them all consistent.
    - MM: no comment.
    - JG: we need one of these. Need a straw poll of the companies.
    - CW: just ship it.
    - DM: the Rust ecosystem converged to not having flags.
    - DP: I slightly prefer #292.
    - JG: #292 is fine.
    - CW: ok, #292 then.
- CW: Please review last couple offline. Damyan in particular.

Agenda for next meeting

- CW: shading language?
- RM: has everyone received my email about uniformity inference analysis?
- CW: received it, but haven't read it yet.
- MM: propose we discuss shading the 29th.
- KR: that's SIGGRAPH week, several of us will be away.
- MM: August 5?
- CW: DN is not available.
- July 22, RM not available.
- CW: put off until everyone's back from vacation? August 19?
- CW: want both Robin and David in the room and because of vacation it's not possible to schedule.
- RM: agree, would like both David and me to be present.
- JG: is there a small slice between the vacations that's not a Monday?
- CW: no, people take weeklong vacations.
- CW: let's plan on August 19. Gives more time to make progress.
- CW: next week people will have thought about bind point renaming. Immutable samplers. Other topics? Snapshot?
- JG: testing.
- CW: we'll have something to show - maybe.
- KN: I think so.
- MM: would like to do some investigation about threading / workers.
- CW: we would like to as well. Have done some investigation internally.