GPU Web 2017-08-23

Chair: Corentin
Scribe: Ken Russell / Corentin Wallez / JF Bastien
Location: Google Hangout

Minutes from last meeting

Tentative agenda

- Administrative stuff (if any)
- Individual design and prototype status
- Shading languages
- Agenda for next meeting

Attendance

- **Dean Jackson (Apple)**
- **Filip Pizlo (Apple)**
- **JF Bastien (Apple)**
- **Keith Miller (Apple)**
- **Myles C. Maxfield (Apple)**
- **Theresa O'Connor (Apple)**
- **Warren Moore (Apple)**
- **David Neto (Google)**
- **John Kessenich (Google)**
- **Kai Ninomiya (Google)**
- **Ken Russell (Google)**
- **Zhenyao Mo (Google)**
- **Ben Constable (Microsoft)**
- **Chas Boyd (Microsoft)**
- **Rafael Cintron (Microsoft)**
- **Dzmitry Malyshau (Mozilla)**
- **Kirill Dmitrenko (Yandex)**
- **Doug Twilleager (ZSpace)**
- **Elviss Strazdiņš**
- **Joshua Groves**

Administrative items

Please please please tell Corentin if you are coming to the Chicago F2F meeting.

Individual design and prototype status

Apple:

- Nothing new to talk about
- Starting to prototype some things but won't be able to talk about it for a week or two

Google:

- Intern Austin Eng finished his internship
    - D3D12 backend is up to par with other backends; only missing one feature
- Corentin wrote doc about pipeline objects and made pull request on gpuweb repo
    - Since we have mostly consensus about pipelines and PSOs, doc aims for more detailed discussions there
    - Ben Constable mentioned an explicit enable/disable depth compare would be useful. Metal has this implicit
    - https://github.com/gpuweb/gpuweb/pull/29

Microsoft:

- No updates

Mozilla:

- Started prototyping based on Servo. Will have results in a few weeks

Shading languages

- CW: Was an interesting discussion via email about SPIR-V, and what it means to define security for shaders
    - Also, whether to make the API use the same language the shaders use
- FP: we're going to have to specify some textual form for shaders to speak about what it means to have a "binding" for WebGPU. Code samples, tests, etc. Something in a human-readable form. So, choosing a binary form of IR here doesn't free us from having to make a source language
- CW: if there's already one for the intermediate language we choose, we can bypass this
- FP: SPIR-V doesn't have a type system sufficient for web security. Unless we want to use the logical addressing mode. Which Apple feels isn't powerful enough. Therefore, the source languages that compile to SPIR-V are insufficient.
- JK: what does (???)
- FP: should be at the level of abstraction of SPIR-V or LLVM IR. And alternate new features to be safe by design.
- JK: what kind of workload are you talking about that logical addressing mode doesn't work for?

- MM: WebGPU will be used for more than graphics. Devs using compute use it in MSL, OpenCL or CUDA. They use C++/ Think WebGPU will be used for ML, AR, CV. Think logical addressing mode or GLSL / HLSL aren't powerful enough to support these use cases. SPIR-V can't go all the way to C++.
- JK: SPIR-V has the spectrum of possibilities. We're adding more pointer support now while keeping them as abstract types. Can do more of this. There is some room in SPIR-V for expanding the functionality without going all the way to C++. Including doing some subset of pointer-type things without having to define how wide pointers are, or doing integer math on pointers. Did a little bit of this already to start supporting OpenCL kernels on Vulkan. Maybe what we are talking about isn't compatible with the security model we are looking for.
- FP: coming from someone who's dealt with type systems for secure CPU-side languages: being able to define syntax of language helps you come up with type system that you're comfortable with. Carefully add only the features you're comfortable that satisfy the security constraints. It's hard to design type systems; consider Java classloader constraint bug, undiscovered at the beginning. We think if we come up with something more complex than GLSL but less complex than C++ will be complex. Not sure that SPIR-V brings any benefit if we have to obey security constraints.
    - FP: The way that compound types are handled is different for example. Usually handled by bitcasting.
    - CW: you're saying that MSL is a good example of a language that supports these use cases, but doesn't it have limitations like no casts?
    - FP: MSL doesn't have the security properties we want. And if you want to add these properties, then you have to change it a lot.
    - FP: If you want pointers to be secure they need to have the bounds with them.
    - DN: pointers on GPUs have always needed to be implemented this way in order to handle robustness extensions.
    - BC: it's not accurate to say that on all GPUs, the level of security exists for pointers. Depending on the driver model, HW, OS you're on, there are various guarantees of robustness.
    - CW: yes, for example SSBOs are sometimes implemented as 1D textures with the sampler doing the robustness
    - BC: out-of-bounds index into array in constant buffer. Reality in a lot of hardware, it's undefined, meaning not secure. Having to carry around the bounds of the array – have to have a story for pointer arithmetic. The hardware won't bounds check for you.
    - JK: it's a question of whether the HW implemented the rules to keep things safe.
    - BC: if only one piece of hardware is able to run the shaders at full speed because the hardware doesn't implement the robustness extensions, that doesn't meet the goals of the project.
    - JK: the input would only run if the platform provides robustness.
    - BC: noone's going to make an insecure implementation of this. On some HW you have UMA. if you can read in-process memory that's a huge security hole. Have

to design this in from the start. Also: Ken has tried to make the point: we're going in circles. We never agreed what the goals were. Talking about how to solve problems, but the problems aren't specified to the degree where we can talk about whether they're solvable or not.
        - So if the goal is that you can write compute shaders with security, then the intermediate format matters a lot.
    - CW: I think that most people agree on what the goals are -- you can't read outside the bounds of the WebGPU device. The question is how that's done.
    - FP: if all pointers to buffers are CPU-style fat pointers, or Cherry-style capabilities with lower and upper bounds, then this is something that needs to be described explicitly in the language, and would be implemented by the languages as a translation step.
    - CW: there are two cases: either the fat pointer style you're describing, or memory enclosure between WebGPU devices by having a different GPU page table per device.
    - FP: we have to talk about performance up front. We can't choose a set of requirements that will kill performance on one platform.
    - JK: Having high-performance in-bounds accesses is a goal.
    - CW: also we can't see another WebGPU device's memory.
    - MM: the design should be more expressive than HLSL and GLSL. Probably shouldn't have general, arbitrary pointers, but should have a way to alias between two buffers.
    - CW: even for a V1: we could limit ourselves to workloads that could be limited to the logical addressing mode, and still get most of what we want out of WebGPU.
    - MM: that's worrying because it changes the shading language decision entirely. Could be a completely different decision than what we choose now.
    - FP: we feel that SPIR-V will significantly limit the long-term "legs" of this spec.
    - DN: Myles mentioned a "feature" which is aliasing (???)
    - FP: in MSL, can pass around a pointer. Return pointers from functions, control flow, etc. This is the sort of thing I'd like us to have. Ideally also some storage of pointers in the heap.
    - CW: Chaz: D3D only has HLSL and not the pointery stuff we are talking about was it a problem?
    - CB: Fro graphics and matrices etc. no. Other code comes from FORTRAN a lot so translates well.
    - MM: that's the point I'm trying to make. Two populations we should be trying to serve: graphics developers happy in GLSL/HLSL. Another population: using GPUs to speed up their existing programs. C++, Swift, etc.
    - CB: Adobe was one of the ones interested in this for portability. Weren't so interested in security. Have both performance critical scenarios where they don't want checking of every pointer, and those that want security. So think we will need both of these.

- FP: why would WebGPU have a layer that doesn't have security? That doesn't seem to be aligned with the goals of the web.
- CB: Devs who are using the Github C++ project that's the foundation of WebGPU might not want to use the safety features of the Web and get more performance.
- MM: think we can all agree that we're making a web API, and any C++ layer underneath is an implementation detail.
- FP: I have no example of a CPU language that is secure and wasn't started with the stated goal to be secure.
- JK: Example workloads that can be done in a secure way but cannot be done in a C way or SPIRV way?
    - FP: Will think of better examples.
- BC: If we are talking about not accessing memory outside of a pointer, we should define what happens when a program does that. We could make it impossible to make programs that do that, or we can add something to the spec that says it returns 0, or in bounds, or crash. We should define that. Should define whether or not it's inexpressible due to static analysis.
- FP: In WASM or JS, we are looking at deterministic immediate failure when something bad happens.
- MM: it's not an option for us: we have a long pipeline.
- ??: we're in agreement that it's hard to specify
- CB: we went through this with D3D.
- MM: Agree but for example is an invocation of the vertex shader does something illegal, should we rasterize the other triangles?
- KR: Previous GPU system built with robustness in mind. In D3D, the DDK …… We can be sure we can't pre-empt kernels. We need to have some kind of clamp / zero behavior. Definitely need testable guarantees. Or static-analysis based guarantees.
- KR: taking SPIRV as an example, we shouldn't discount proposals but expose issues and get inspiration from other secure languages like WASM. We need to understand their designs, and see how they could happen on GPU. For example GPUs don't have signals so the WASM thing of relying on boundary pages(?) wouldn't work on GPUs. Need to expose what previous solutions have been.
    - Having specific implementations penalized 50% not okay
    - Exposing random bits of video memory not okay
- KR: On the SPIRV side we should look at what logical mode gives us, what the variable pointer extension gives etc. On the WASM side could you provide pointers to the design?
- JFB: see PLDI paper on WebAssembly. Touches on some of this. Please ping JFB for more detail on anything the paper is missing to answer this question. https://github.com/WebAssembly/spec/blob/master/papers/pldi2017.pdf
- FP: something to appreciate is taking something with SPIR-V semantics and making it secure is hard. C-Cured and Cyclone. Seeing how far-reaching the changes to the language are.

- DN: Please send references to this group / ML.
- KR: There was a number of iterations to SPIR-V that we shouldn't dismiss, for example the logical addressing mode if much better than anything there was before, like OpenCL that put the onus on the implementation to do fat pointers.
- JK: the logical addressing mode is a significant constraint, but shouldn't discount the fact that pointers can be abstract in SPIR-V, so there is room to grow. It has done something different in this space. Don't think we're going to take arbitrary C workloads and start securing them. Need to understand what's the actual functionality gap between what we want to run, and what can be run with the normal GPU shading model.
- KM: does it make sense for people to go find the workloads they intend to write, and see if they're reasonable?
- MM: We'll do that on Apple's side.
- JK: for example, do you need to be able to take the address of an object, treat it as an integer, mask it, use it as another pointer, etc.?
- MM: Can do it.
- MM: not trying to dismiss SPIR-V outright. Open to all possible way
- KR: Let's try to enumerate more of the design space. Examples of system with static analysis-based security would be good to.
- KM: ideally: whatever's cheap or free to do statically. Then whatever you can't, do at runtime.
- JFB: FWIW this is what we do in WebAssembly. Try to make all checks zero-cost, but sometimes inline them in the code. Could eliminate some of them, but in JSC we don't because this is the uncommon case.
- FP: in a highly optimized Java implementation, the bounds checks cost 5-10% total. So there's an existence proof that you can have security guarantees plus high performance when you're careful about how you expose arrays and pointers.
- DN: static checks will still happen when the program's downloaded. Need them to be cheap.
- KR: <amazing joke on the halting problem>
- CB: do we want to have some sort of ascii representations that people can use as an exchange format? If we only have an IR then we'll have fifteen different shader languages. Would like to at least have all implementations to have one agreed-upon supported syntax
- MM: Agreed, hostirocally, each graphics API has one dominant human-authored shading language. Don't see why WebGPU would have to be different. The human-authored shading language should be defined in this group.
- JK: this is changing with OpenGL and Vulkan. Because there are these intermediate representations, GLSL and HLSL are more or less on equal footing. No longer one dominant language.

    - KM: a human-readable syntax is useful for people who want to write tests, too. Was useful for WebAssembly. There's now a text format for WebAssembly that was created by the WG.
    - CB: consider requirement that implementations include such a compiler. Just to avoid the fragmentation.
    - MM: shouldn't define two languages.
    - CB: if we don't define an ASCII syntax then we'll have a lot of divergence.
    - FP: should also consider only specifying a textual language, ???
    - CB: Should also provide a machine version of the specification of this thing.
    - KR: The WebGL community has encoded things through sweat and blood in the CTS. There's insane amounts of workarounds so WebGL is robust when passed to high-level GLSL / HLSL compilers.
    - DJ: Why is a binary format better than a source format? We are going to translate to the low-level compiler anyway.
    - JK: It wasn't binary vs. textual, but that the form of the language is very rigid. There shouldn't be a way to represent things in many different ways and a lot of things done for the programmers.
    - FP: In JVM world we had security bugs when changing the binary format so binary doesn't necessarily help.
    - KR: my point was that having an IR where for example all of the values have been lowered into SSA form is a great help to eliminate problems like where compilers get the scoping and shadowing rules of the language and spec wrong.

- CW: list of action items and items to discuss.
    - Pointers into WASM and secure CPU languages
    - Define the goals of what workloads we want to run
    - Blessed human-readable language?
    - Define what security means
    - SPIRV folks to look into security / perf of logical addressing mode and variable pointer extension.
    - Binary vs. textual, intermediate vs. high level.
- KM: would be good but not required to have actual code examples.
- MM: will be trying to bring some of these. Also will be trying to play around with our findings and maybe come up with some strawman implementations / languages.

Agenda for next meeting

- CW: Is a week anough to discuss all this or do we need more?
- MM: Two weeks would be good.
- Everyone agrees, meeting in two weeks about shading lnaguages.
- CW: Will send an email to decide topics for next week on the ML.