GPU Web 2017-06-07

Chair: Corentin Wallez & Dean Jackson
Scribe: Dean (with help)
Location: Google Hangout

Minutes from last meeting

Tentative agenda

- Administrative stuff (if any)
- Individual design and prototype status
- Base explicit API concepts (pipelines, command buffers, queues)
- Other topics we didn't get to last meeting
    - Render-targets / render passes
    - Stuff that doesn't go in (XFB?)
    - How to reach consensus on WebVulkan vs the alternative
- Agenda for next meeting

Attendance

- **Chris Marrin (Apple)**
- **Dean Jackson (Apple)**
- **Jason Aftosmis (Apple)**
- **Julien Chaintron (Apple)**
- **Myles C. Maxfield (Apple)**
- **Theresa O'Connor (Apple)**
- **Gareth Morgan (Axum Graphics)**
- **Austin Eng (Google)**
- **Corentin Wallez (Google)**
- **Kai Ninomiya (Google)**
- **Ken Russell (Google)**
- **Ricardo Cabello (Google)**
- **Zhenyao Mo (Google)**
- **Daniel Johnston (Intel)**
- **Ben Constable (Microsoft)**
- **Rafael Cintron (Microsoft)**
- **Dzmitry Malyshau (Mozilla)**
- **Jeff Gilbert (Mozilla)**
- **Kirill Dmitrenko (Yandex)**

- **Doug Twilleager (ZSpace)**
- **Elviss Strazdiņš**
- **Joshua Groves**
- **Tyler Larson**

Administrative items

- None

Individual design and prototype status

- CW: What are members up to with their prototype?
- CW: We've kept prototyping NXT. We have tests running on precommit and the validation code, but nothing actually on the GPU. We've started on RenderPasses, starting on DepthStencil and mapping buffer for reading.
- CW: Austin has started on the D3D12 backend got HelloTriangle.
- DM: No update on Mozilla's prototype. Given the recent discussion on binding models, I'm hoping we can investigate some of the decisions made on Issue 19 to see if they are valid.
- JG: I started writing a skeleton for Vulkan to D3D12.
- DJ: Nothing much on Apple's WebGPU. Will look at changing the API to look a bit more like Vulkan with descriptor sets.
- JG: Will you change anything because of the introduction of Metal 2?
- DJ: Metal2 available on High Sierra, so it would only work for people on the Beta. While Metal2 would have advantages, we might not use it for a while.
- BC: any resources on Metal 2?
    - DJ: look at WWDC videos, maybe 5-10 min segment on indirect argument buffers
    - DM: The Metal2 slides
- CW: if you want to look at indirect argument buffers, there's a link in the binding model issue #19 on Github that point to the Metal docs.

Pipeline state

- CW: Let's talk about the basic stuff that isn't the binding model.
- CW: All APIs have pipeline states that are separated between graphics and compute. How much do we want to put in there?
- JG: What are the notable differences?
- CW: D3D12 can do things like binding the depth stencil state outside the pipeline, but Vulkan is inside. Overall, the Vulkan pipeline state is bigger than the others. This means we'll have to decide if we're ok with having to recompile on Vulkan.
- DM: The cache objects in Vulkan should make that less of an impact.

- CW: Do people have an issue with having a fat pipeline state, with all the pipeline aspects that any of the native APIs has in pipeline state?
- DJ: I'd like to know what real world impact is.
- KR: Thinner states allow the application to toggle back and forth.
- JG: We don't have a lot of data on this, because we don't have content. If any of the ISV/IHVs have information, we'd like to hear it.
- DM: I think there is some info in the Vulkan specification saying that two similar states will efficiently compile and swap.
- CW: It should be cheap to do a check on setPipeline to see if it is different.
- DM: I'm not sure I agree that the Vulkan pipeline is bigger than D3D12, because the render target is embedded in the pipeline.
- CW: Is the render target view included in the D3D12 pipeline state?
- DM: Yes, it is.
- BC: At Microsoft, we had a lot of discussions as to what is in the pipeline state. We should do a side-by-side analysis here for all APIs. I was hearing that people want WebGPU to be the aggregate of all things.
- Elviss: on the Web 2D games are popular and they change blend state a lot so pipeline state so it would be a lot of pipeline state.
- Rafael: In D3D12 you specify the format of the render targets not the RTV itself
- JG: That's included in the renderpass that you give on the Vulkan pipeline state.
- CM: I want to comment on "if two states are similar, it should be efficient to swap between them". I don't think we can know this for sure. It might be rendering library and OS dependent.
- JG: I think we should do a direct comparison between all three APIs, and note the differences.
- CM: There are also some pipeline state attributes that are optional in the Vulkan pipeline state for GPUs e.g. scissoring.

ACTION: JG to write up a comparison of pipeline states between Vulkan, Metal and D3D12

Command Buffers

- CW: Metal has different kinds of command encoders
- CW: Secondary command buffers in Vulkan
- JG: Need to compare/contrast secondary command buffers with D3D command bundles
- Rafael: Bundle restrictions
- DM: The problem is that there isn't a single solution that's better than others. Vulkan secondary command buffers are powerful but d3d12 bundles are more efficient. Even if we came up with a model that includes both, we'd sacrifice some performance.
- JG: It would be nice to get some examples of what is and isn't fast between the two. I expect this might be a micro-optimization that doesn't matter too much.
- MM: So far we haven't discussed primary command buffers, so I'm not sure we need to decide on secondary buffer yet. Not necessary for an MVP.

- BC: I agree with that sentiment. It doesn't need to be there for bootstrapping. There are other crucial parts.
- CW: So we should defer discussion on secondary command buffers?
- BC: We don't have the feature-set of the MVP, and it would be more focused than wondering about secondary command buffer / bundles
- KR: Secondary Command Buffers were added to Vulkan specifically for mobile GPU performance. <Mistake: was thinking about render passes>
- BC: It's still an interesting point - do passes need to be part of an MVP for performance?
- CW: I think that is a big topic that we should discuss later.
- DM: I think Ken's point is important. Secondary Command Buffers were added to Vulkan because of Render Passes. They are related issues.
- CW: Let's talk about it when we discuss Render Passes.
- CW: The biggest API difference is that Metal has multiple encoders. How expensive is it to swap?
- MM: If you're trying to emulate one big pipeline, it's setting up the state for all the previous passes. (bad minuting)
- CW: Vulkan prevents you from …
- MM: If the semantic model of Metal is the same as the semantic model of Vulkan, but described in a different way...
- CW: the biggest difference between Metal and Vulkan seems to be whether compute and blit can be interleaved
    - Do we need to reset state between compute operations?
- BC: in D3D12, there's a queue for graphics, blit, compute (clarification: direct, bundle, queue, copy)
- CW: Can't you have a queue with everything?
- JG: In Vulkan these are capabilities bitfield on a queue
- BC: In D3D12 you have a choice of direct, bundle, copy, or compute queue.
- MM: So since D3D12 and Metal both have this concept of describing the type of queue, and Vulkan can handle that, I think it should be the model.
- JG: That is the way I was approaching it in my prototype
- BC: I will investigate about supporting queues that have different operations.
- CW: Vulkan does have the concept of a queue that supports all types of operations.
- CW: Should we figure out what exactly happens on all three APIs, and go for the minimum supported semantic. That way we don't have to switch too much on the lowest common denominator.
- BC: I agree. Having to do lots of work underneath will be too complex.
- DM: Two different questions - what the encoder provides and what the queue provides. Given Apple's approach is more explicit and can provide more type safety, it sounds like the right approach.
- BC: Is an "encoder" a "command list" in D3D12?
- DM: No. It is a context in which you populate your command list.
- JG: A command factory.

- MM: you create encoders from command lists to call the operations on. The encoder records the calls in the command list.
- BC: In D3D12 you have the command list that does the recording, and then is closed before it can be sent to the GPU. There is a CommandAllocator that is basically is a memory allocator.
- JG: Metal encoders look like an RAII for BeginCommandBuffer and EndCommandBuffer.
- CW: What happens when you switch encoders? If it is just a CPU API convenience then is it cheap to switch them?
- DJ: Sounds about right.
- DM: If you do render / blit / render (with the same render target) what happens?
- CW: <something about rendercommandencoder being like renderpass for restrictions>
- DJ: someone should do an investigation on this, and look at which commands are allowed where.

ACTION: CW will start an issue about this, and people from different APIs can fill in the blanks for their system.

Command Queues

- CW: How important are async compute and async blits? AMD has reported +15% perf increases via async compute.
- JG: keep queues; doing a polyfill on top of Metal is not worse than not having them.
- BC: background on why there aren't multiple queues in Metal? Usually HW has multiple HW queues. What's the reasoning behind the decision in Metal (or am I interpreting it wrong?)
- DJ: will have to ask
- BC: Want to make sure we aren't running against the flow of how Metal is going to evolve in the future.
- JG: Is Metal internally or externally synchronized? Is calling queueSubmit on multiple threads at the same time ok?
- DJ: Yes
- JG: In Vk you need to synchronize queues yourself, allows to use them on multiple threads (and reduce overhead)
- CM: I'd like to understand the concept of queues between all three APIs. Are they used differently?
- BC: In D3D12 you can have a copy only queue that doesn't use the 3D cores for the copy. You can use the queues concurrently and you have fences to synchronize them. Everything is done async so fence to synchronization at a distance. HW might be able to process stuff in parallel.
- DJ: my guess is that a blit encoder can run stuff in parallel with compute encoders.
- BC: In D3D12, fences are done on queues not command lists
- CW: same in Vulkan and Metal.
- JG: In Metal you get a callback.

- CW/DC: Can’t synchronize multiple queues in Metal.

ACTION: JG to write up a discussion on the queues between the three APIs.

Agenda for next meeting

- DJ: Agenda for next meeting should include talking about the investigation that were made.
- DJ: W3C has a technical meetup (F2F, next to SFO), should we meet? WebAssembly is there would be interesting to meet.
- DM: We are already in the agenda
- DJ: Will send details to the ML.
- CW: also for next meeting’s agenda, if we have more time after the retrospectives, we could talk about renderpasses.