NXT “Memory barriers”

cwallez@google.com, kainino@google.com

API analysis on the gpuweb issue tracker

TL;DR of the analysis

What we call “memory barriers” regroups two things in hardware:
- Memory hazards where two parts of the GPU pipeline access the same memory. This can even be hazards of a stage with itself.
- Image swizzling and transitions between different types of swizzling. Called “image layout transitions” in Vulkan.

The different native APIs provide the following controls:
- D3D12: at any time, each resource can be in one writable usage or a combination of read-only usage. The API provides a way to transition a subresource between usages, causing both synchronization to avoid memory hazard, and swizzling changes to occur.
- Metal: everything is implicit, but the driver doesn’t emit “memory barriers” inside MTLRenderEncoders (only between render encoders).
- Vulkan: A vkPipelineBarrier allows specifying synchronization for memory hazards per-resource or globally. The same command independently specifies image swizzling transitions. Some implicit swizzling transitions happen at renderpass boundary.

Usage transitions in NXT

Error management is briefly mentioned but we are intentionally not detailing it here. As usual, please don’t mind the cosmetics of the API.

Overview

For WebGPU we were looking for an implementation of “memory barriers” that is efficient to validate while still providing most of the control D3D12 and Vulkan provide. We settled on