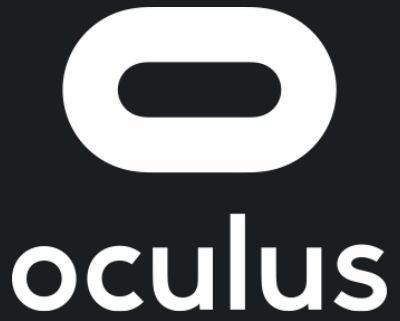

WebVR Next w/ More Layers

Web VR Performance 10, 11… A lot of steps ;-)

Web VR 1.x Core Goals

- Avoid UA Specific Details that would hold up Adoption
- Avoid Device Specific Details that would hold up Adoption
- Focus on Primitive VR Display Access and existing WebGL Functionality
- Avoid Crossing Spec Streams
  - Ex: We didn’t specify HTML 5 Event Loop changes for new requestAnimationFrame
  - Ex: We didn’t make Gamepad updates for motion controllers a requirement

Web VR Next Core Goals (Potentially)

- Tackle UA Details Where Perf/User Experience are Greatly Impacted
- Broaden Device Understanding and Clarify Spec where Possible
- Add More VR/WebGL Performance Primitives
- Add Some Support for Device Specific Layers
- Cross the Streams – Web Application Manifest, Service Worker, WebGL, Web Imaging, Gamepad, Workers, etc…

- Some Incremental + Some Reserved for a Major Breaking Change

WebGL Capabilities

- Multi-view, stereo instancing, any overhead reducing tech
  - Escape hatch here, since there might be engine side optimizations possible
- Geometry Tools –
  - Fonts
  - HTML 2 Geometry (instead of just texture)
- HTML 2 Texture (also useful for Canvas, enabled by?)
  - Reduces overhead from localized texture download
  - Allows dynamic content
- Compute Shaders – Some things are still not efficient in the CPU

Retained Mode Capabilities

- Layer Specification syntax for requestPresent
  - Persistent, static layer, useful for loading screens (world locked)
    - Can be made simply animated as in the Oculus SDK (rotation)
    - Can mostly be done with a single submitFrame() but needs hinting for best experience
  - Cursor layer – Avoids the double cursor effect during frame skips (head locked)
  - Cylinder layers for high quality 2D surface presentation
  - Cube Map layer (combined with Cylinder == complete scene)
  - Going too far is demons – Should not be a full retained scene graph
- Asynchronous Layer Enable/Disable Controls (useful?)

Execution Model (Threading, GC, Run-Time)

- Existing HTML 5 Event Loop Optimized for 2D, HTML Content
- Reprioritization of Event Loop Tasks
  - Downgrade layout, idle tasks
- Garbage Collection – Timesliced Finalization, More off-thread
- VR Only Threads and Workers??
  - Singleton with Custom Global (VRWorkerGlobalScope)
  - Access to HMD and submitFrame off of Main Thread
  - Worker all the things!?! (Gamepad, Haptics, Audio)
  - Maybe Rental threaded Worklets instead… Unsure here.

Various APIs (Time Permitting)

- Finish Image Loading APIs
  - ImageBitmap and createImageBitmap (ensure off-thread)
- Figure out Compressed Textures
  - Resolves the Client Optimization phase in the Content Pipeline
  - toDataUrl() or something new
- Update Core HTML 5 Navigation Model for VR to VR Linking
  - Will eventually improve responsiveness of initial paint
- Crazy Stuff
  - Cross Navigation GPU Resource Usage/Caching (can be done in the UA)
  - Browser Supplied Layers – Hands, System UX, etc…