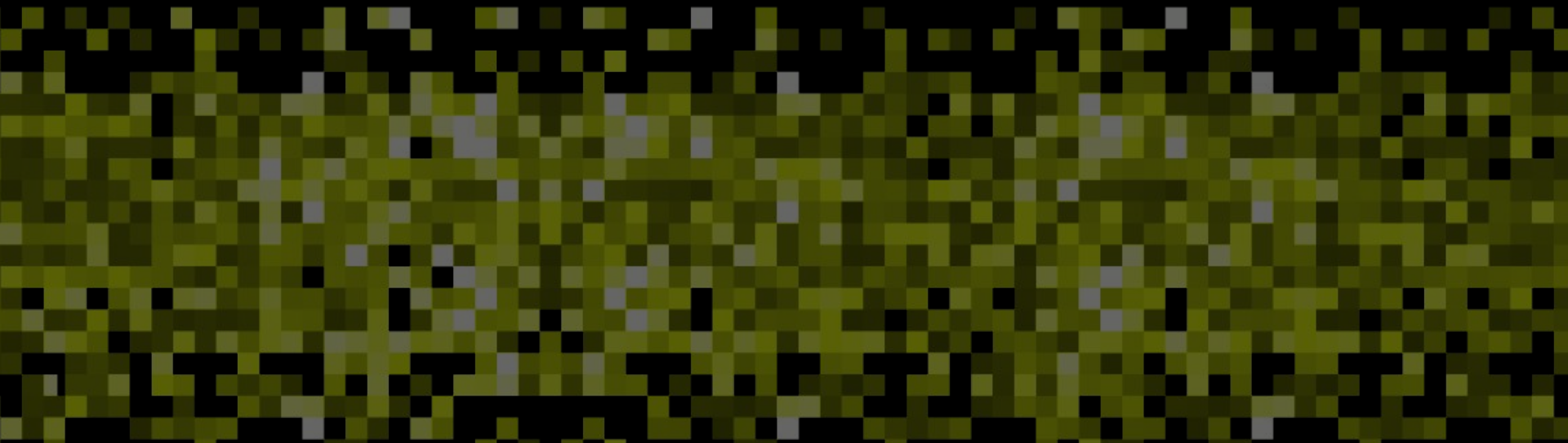
moz://a

Challenges of the VR Web



What is WebVR?

- Low level API to interface with VR hardware.
- **Input**: Access to headset and controllers pose.
- **Output:** Efficient and versatile mechanism to send pixels to the headset display (high FPS and low latency).

WebVR 1.1

Go to the spec.

1. Enumerate VR displays.
2. Configure layers (WebGL sources).
3. Request a VR display animation frame.
4. Get frame info:

   view and projection matrices per eye.
5. Draw the scene (once per eye).

WebVR 2.0

Go to the explainer.

- Feature-support test.
- `vrpresent` new context for canvas:

  mirroring & magic window
- Multiple frame of references:

  headModel, eyeLevel, stage.
- Supports of stage bounds.
- Multiview support.
- Drops resetPose()

moz://a

Possible to soften DEPRECATED language in the 1.1 spec? #288

State of WebVR

*

**

* Vive and Oculus
** Under Origin Trial

WEBVR 1.1

Open Challenges

- ➔ VR Navigation
- ➔ Trusted UIs
- ➔ DOM to texture
- ➔ AR conversations
- ➔ Rendering pipeline optimisation

VR Navigation

Browse, navigate, explore…

Links and seamless navigation between pieces of content is one of the core ingredients of the Web. **What about WebVR?**

1. window.addEventListener('vrdisplayactivate', onChange)
   - Monitor changes on the session
2. function onChange(evt, vrDisplay) { if (evt.reason === 'navigation') continueSession(vrDisplay); }
   - Not actually needed because...
3. function continueSession(vrDisplay) { vrDisplay.requestPresent(); }
   - requestPresent() won't fail if already presenting (even out of a user gesture)

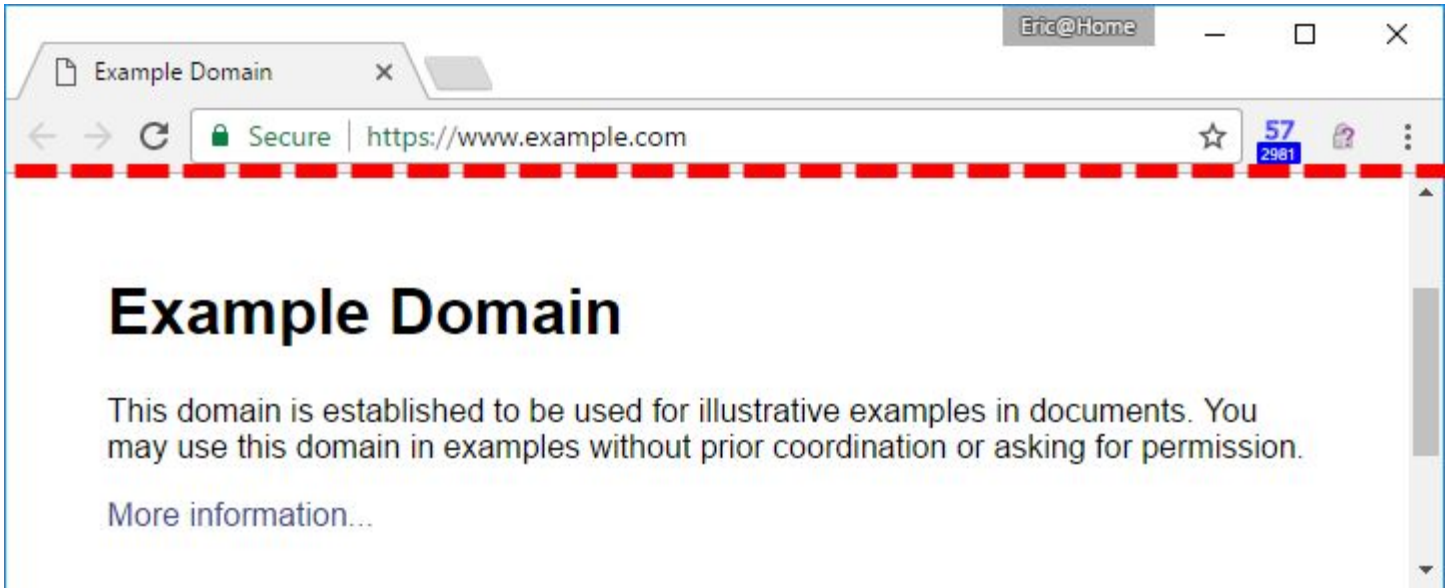
Open challenges:

➔ What happens in the middle? While loading the target site.
➔ How does the user distinguish they jumped to another experience?

Trusted UIs

The problem of recreating **line of death** in VR:

- ➔ What about the URL bar? How can I get the current address?
- ➔ And permission prompts like getUserMedia() or geolocation?
- ➔ How can we prevent content from simulating the UA?
- ➔ Error/status messages: 404/No network
- ➔ Fallback for unresponsive websites



DOM to texture

Browse the traditional Web in VR and **use the DOM to build 2D UIs in 3D.**

Zhenyao Mo (Google) presented a prototype at Siggraph 2017:

1. WebGL.bindTexSource(GLenum texTarget, GLint texLevel, GLenum texFormat, GLenum texType, HTMLIFrameElement iframe)
   - Bind an Iframe to a texture, the texture is dynamic and read-only, only call once
2. WebGL.requestFrame(GLenum target)
   - Ask the rendering engine to produce the IFrame rendering result to the texture, call every frame
3. Canvas.setEventDispatcher(function(coordinates){}, iframe)
   - Only WebGL app knows if the input event is targeted at the IFrame and how to transform the coordinates
   - Concerns: app may manipulate this (for example, user click “No”, app send out “Yes”)

Rendering pipeline optimisations

- VR is very performance demanding:
  - **Monitor:** 1080p @ 60fps = **124M** pixels/second
  - **VR (HTC Vive):** 1512x1680x2 @ 90fps = **457M** pixels/second
- Goal #1: Avoid duplicating a complete render pass for each eye:
  - From: forEach **camera** -> forEach **object** -> render
  - To: forEach **object** -> forEach **camera** -> render
- Improving browser VR support:
  - **Multiview:** Reduce drawcalls by half (CPU bound)
  - **Multi-res shading / Lens Matched shading:** Reduce fragments to process (GPU bound)
  - **WebGPU** + WebASM

AR conversations

Both Mozilla and Google are **exploring AR extensions** to the WebVR spec. Unoficially called WebXR.

- ➔ Google has WebAROnARCore & WebAROnARKit
- ➔ Mozilla has an ongoing WebXR on iOS and a not-yet-ready WebXR spec.

@mozillavr

@dmarcos
@fernandojsg
@salvadelapuente

https://vr.mozilla.com

See you on the other side!