Reaching the Largest Gaming Platform of All:
The Web. WebGL, WebVR and glTF
GDC, February 2017

Neil Trevett
Vice President Developer Ecosystem, NVIDIA | President, Khronos | glTF Chair
ntrevett@nvidia.com | @neilt3d
Khronos News Here at GDC 2017

• WebGL™ 2.0 Specification Finalized and Shipping
  - https://www.khronos.org/blog/webgl-2.0-arrives

• Developer preview on glTF™ 2.0
  - https://www.khronos.org/blog/call-for-feedback-on-gltf-2.0

• Announcing OpenXR™ for Portable Virtual Reality
  - https://www.khronos.org/blog/the-openxr-working-group-is-here

• Call for Participation in the 3D Portability Exploratory Group
  - A native API for rendering code that can run efficiently over Vulkan, DX12 and Metal  _khronos.org/3dportability_

• Adoption Grows for Vulkan®; New Features Released
  - Details here
OpenXR - Portable Virtual Reality

[Diagram showing the integration of different applications, engines, and devices through the OpenXR Application Interface and Device Layer.]
OpenXR Details

OpenXR Working Group Members
Design work has started in December
Estimate 12-18 months to release

WebVR would use WebGL and OpenXR
3D Portability API - Call For Participation

A Portability Solution needs to address APIs and shading languages.

‘WebGL Next’
- Lifts ‘Portability API’ to JavaScript and WebAssembly
- Provides foundational graphics and GPU compute for the Web

‘Portability Solution’
Portability API Spec
+ Shading Language open source tools

API Overlap Analysis

Open source compilers/ translators for shading and intermediate languages

GLSL
HLSL
MIR
MSL
DX IL
GLSL
HLSL

© Copyright Khronos Group 2017 - Page 5
### Speakers and Topics for This Session

<table>
<thead>
<tr>
<th>Company</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>WebGL 2.0</td>
</tr>
<tr>
<td>Zhenyao Mo, Kai Ninomiya, Ken Russell</td>
<td></td>
</tr>
<tr>
<td>Three.js</td>
<td>Three.js</td>
</tr>
<tr>
<td>Ricardo Cabello (Mr.doob)</td>
<td></td>
</tr>
<tr>
<td>Google</td>
<td>WebVR</td>
</tr>
<tr>
<td>Brandon Jones</td>
<td></td>
</tr>
<tr>
<td>NVIDIA</td>
<td>glTF</td>
</tr>
<tr>
<td>- Neil Trevett</td>
<td></td>
</tr>
<tr>
<td>Microsoft</td>
<td>Q&amp;A</td>
</tr>
<tr>
<td>- Saurabh Bhatia</td>
<td></td>
</tr>
</tbody>
</table>
WebGL 2.0

Zhenyao Mo, Kai Ninomiya, Ken Russell
Google, Inc.

Khronos Developer Day
Reaching the Largest Gaming Platform of All: The Web.
WebGL, WebVR and glTF
Background on WebGL

WebGL is being used for all kinds of 3D visualization and presentation on the web today.

- Artists showcase their work on Sketchfab, with over 1 million models shared and embedded across the web.
- The New York Times publishes visualizations in Three.js
  - Jupiter and Its Moons
  - The Dawn Wall, El Capitan’s Most Unwelcoming Route

Create amazing, fast, and interactive mathematical visualizations using MathBox
WebGL in Games

Everyone has a web browser
WebGL “just works” for almost everyone!
WebGL Stats - 96%
caniuse - 91%

Unity - HTML5 target
Publish Unity games straight to the Web with HTML5 and WebGL

Open-source, pure-web 3D and game engines
PlayCanvas, Turbulenz, BabylonJS, Three.js
WebGL-accelerated 2D game engines

…
WebGL 2.0 + WebVR

(Github repository, Live Demo)
PlayCanvas: After the Flood [Video]
PlayCanvas: After the Flood

Techniques used
HDR+MSAA rendering, using z-buffer as a texture, procedural clouds using 3D textures, hardware PCF, alpha to coverage and transform feedback particles.

Demo features
procedural clouds, procedural water ripples and reflection, leaf particles, animated trees, mirrors, dynamic lights, compressed textures, asset streaming, runtime lightmap baking, and more.
WebGL 2.0: Feature Overview

WebGL 2.0 exposes the OpenGL ES 3.0 feature set
Brings desktop and mobile platforms’ features much closer together
Features (details later)
Many sized texture formats
   integer/float textures
3D textures, 2D texture arrays
Immutable textures
Full non-power-of-two texture support
Instanced drawing*

Multiple render targets*
Transform feedback
True integer vertex attributes
Multisampled renderbuffers
Many shading language upgrades
   texture level-of-detail sampling control
   (important for physically-based rendering)
   uniform blocks
   in/out variables instead of “attribute”, etc.
   control over layout in the shader

*Available via extensions in WebGL 1.
WebGL 2.0: Feature Overview (continued)

Features (details later)

Seamless cubemaps
  Important for physically based rendering pipelines; can finally use mipmap generation effectively

Performant GPU-side copy/compute operations
  Upload textures from pixel unpack buffers, read pixels into pixel pack buffers, copying between buffers
  Transform feedback
    When used in conjunction with 3D textures and 2D texture arrays, have much more computation ability than WebGL 1.0

Uniform buffers, Vertex array objects*, Sampler objects
Query objects, Sync objects

*Available via extensions in WebGL 1.0
WebGL 2.0 is here!

WebGL 2.0 now available in Chrome/Firefox!

- Chrome: Released to desktop platforms - soon on Android
- Firefox: Released on all platforms
- Edge/Safari: plan to ship WebGL 2.0
- Desktop support at 35% - to continue rising [WebGL Stats]

WebGL 2.0.0 conformance updates

Test suite on track for ratification

- Very thorough - run 10x longer than WebGL 1.0 tests

Chrome

- Windows/macOS/Linux: Passing 100% in Chrome 56 (January)
- Android: Passing 100% in Chrome 58 (April)

Firefox

- Windows/macOS/Linux/Android pending; very nearly conformant
Textures: More Formats

Floating-point textures, unsigned/signed integer textures

A shader can read from and write to a texture with accurate values

- No losing precision, no clamping (no need to worry about range)
- Deferred rendering, scientific computation, etc

8bit, 16bit, 32bit signed/unsigned integer formats; 16bit, 32bit floating-point formats

Demo: Integer textures for RNG in Monte-Carlo simulations [Evgeny Demidov]
Samples Pack: simple integer texture demo

sRGB formats

- Higher quality rendering in linear color space
- Essential for high dynamic range rendering [Floored]
Samples Pack: simple sRGB texture demo

ETC2/EAC compressed textures via extension

(Removed from core spec due to lack of hardware support on desktop platforms.)
Textures: 3D Textures

2D texture arrays & 3D textures

- Volumetric effects (fire, smoke, fog, etc)
- Medical imagery (MRI, CT scans, etc)
- Cached lighting effects

NVIDIA’s smoke box demo

Has been emulated before in WebGL 1.0, for example in Vicomtech’s demos, but now supported natively

Samples Pack:

- Texture array demo
- 3D texture demo
Textures: Immutable textures

Dimensions/format/type will not change once initialized

Better performance - driver optimizations
When implementing WebGL on top of Direct3D, saves a lot of CPU memory

Direct3D doesn’t allow level-by-level texture uploading
A copy of all levels have to be kept in CPU for mutable textures

Demo: Flat Wave (immutable RG32F) [Evgeny Demidov]
Samples Pack: simple demo
Textures: Others

Full support for non-power-of-two textures
  Filtering, wrapping, and mipmapping of non-power-of-two textures
  Long-requested feature; now it’s portable across devices

Seamless cube maps
  Used in physically based rendering pipelines
  Now supported natively
Instancing

To draw 1000 soldiers

WebGL 1: **1000** draw calls, with different position, posture uniforms, etc.
WebGL 2: Only **1** instanced draw call
Big performance improvement

Demos:

Three.js demos:
- instancing demo (single triangle)
- indexed instancing (single box), interleaved buffers, dynamic updates

Brandon’s first demo of the extension:
- WebGL instancing with ANGLE_instanced_arrays

Crowd demo [Github] (image)

Samples Pack: simple demo

Google Maps’ 3D mode uses instancing support
Reduces vertex buffer size by 6x (!) - thanks aappleby@ for the info
Multiple Render Targets

Allow rendering to multiple framebuffers in one pass
Critical for deferred shading

- Lighting calculation on the entire scene once, not per individual object
- Render depth, color, normals, etc into multiple buffers
- Render one full-screen quad which performs all lighting calculations per-pixel
- Big performance gain

Demos

- WebGL Deferred Shading [Sijie Tian, etc.] (image)
- Deferred Irradiance Volumes [Florian Bösch]
- Deferred Rendering [m_panknin]
- Spiral waves in excitable media [Evgeny Demidov]
Integer vertex attributes

Go hand-in-hand with transform feedback, as well as integer texture support

Examples
  - Maintain a pseudo-random number generator’s state per-vertex

More generally:
  - Can send multiple integers into the vertex shader
  - Perform logical operations on them
  - Output them
  - More?
Transform feedback

Save the output of the vertex shader directly into a vertex buffer object
Optionally skipping rasterization

Caching and reusing GPU computation

Crowd demo [Github]
Instanced rendering of many skinned object
Skinning computations for just the few variants, instead of per-instance

Stateful particle systems

Interactive particle demo [Github]
Per-particle state, particle birth/death
Simulates physics each frame

Lorenz strange attractor [Evgeny Demidov] (image)
Samples Pack: simple demo

Non-web OpenGL examples:

OpenGL 4.0 review [Christophe Riccio], Noise-Based Particles advection shader [Philip Rideout]
Multisampled renderbuffers

Efficient antialiased offscreen rendering
User controls number of samples
Useful for implementing techniques like vector curve rendering

Three.js vector curve rendering (image)
Rendering SVG paths in WebGL
Samples Pack: simple demo
Performant GPU-side copy/compute operations

Copies between textures, pixel buffer objects
Transform feedback
Rendering to 2D texture arrays and 3D textures
  Rendering to slices of 3D textures [Evgeny Demidov]
Samples Pack: rendering to texture array
Many kinds of volume rendering algorithms now possible in WebGL
With 3D textures and 2D texture arrays, 3D GPU fluid simulations
Features for performance optimization

Uniform buffers (potentially significant speedups)
  Allows updating many uniforms in a batch with one API call
  Samples Pack: [simple demo](#)

Vertex array objects (also a WebGL 1.0 extension)
  Encapsulates all data that is associated with the vertex processor
    Instead of the actual data, it holds references to the vertex buffers, index buffer, and layout specification
    Usually set up once, and then just keep using it

Occlusion queries
  Used to determine if a render will be visible
    E.g. occlusion-query a bounding box to see if it’s visible, for culling
  Samples Pack: [simple demo](#)
Sampler objects

Stores the sampling parameters for texture access inside of a shader
Use N samplers with M textures - N+M objects instead of N*M
Samples Pack: simple demo (image)
  One texture with 4 different samplers
GLSL ES 3.00: #version 300 es

Samples Pack:

- Texture lookup in vertex shader, e.g. terrain, bump mapping (image)
- Centroid interpolation
- Fragment discard
- Flat/smooth interpolators
- Non-square matrices
- Sampling textures by LOD
- ...

…
And more!

There are more features not listed here
And many new effects not yet implemented

Anything that’s possible with OpenGL ES 3.0

Terrain Rendering with Geometry Clipmaps [ARM Mali Developer Center]
...

Dive in with the WebGL 2.0 Spec
Check out WebGL 2 Samples Pack for minimal examples
Looking Forward

What’s in the future?

Compute shader extension
For advanced effects and GPGPU computation
Parallel data structures & algorithms
Physics simulations
Ray tracing
AI & neural networks
...

A next-generation web graphics API
Widespread interest from developers and browsers to create a new graphics API
Pre-validated design to reduce draw call overhead
Implementable with excellent performance on Vulkan, Metal, and D3D12
Secure design for the Web
JavaScript & WebAssembly
Conclusion

The amazing power of the graphics processor is available to web developers

WebGL 2.0 provides a much needed upgrade to the OpenGL ES 3.0 feature set

Will enable more amazing applications to be built

Now we need you:

What will you create with these graphics capabilities?
Many thanks to the many WebGL 2.0 contributors

Khronos Group

ANGLE team
  Geoff Lang, Jamie Madill, Corentin Wallez,
  Shannon Woods

NVIDIA Mobile Graphics Team
  Olli Etuaho, Kimmo Kinnunen, Amal Prabhu,
  Barthold Lichtenbelt

Intel Web GPU Team
  Yunchao He, Qiankun Miao, Yang Gu, Xinghua
  Cao, Jiawei Shao, Yizhou Jiang, Guanxian
  Li, Chenglei Ren

Firefox team
  Jeff Gilbert

Chrome GPU team
  incl. Brandon Jones, Victor Miura

WebGL working group members
  Mark Callow, Rafael Cintron (Microsoft), Dean
  Jackson (Apple)

Mobica (Janusz Sobczak and team)

Unity
  Jonas Echterhoff, Christophe Riccio, Marco
  Trivellato, ...

WebGL2Samples Pack team
  Patrick Cozzi, Trung Le, Shuai Shao, University
  of Pennsylvania

PlayCanvas, Sketchfab, Floored, Google Maps

The Three.js community
  esp. Ricardo Cabello

...and many more collaborators in the open-source
  community

Authors of linked demos
  Alec Miller, Figma
  Evgeny Demidov
  Florian Bösch
  m_panknin
  Sijie Tian
  ...

...
Appendix: WebGL 2.0 Feature List

OpenGL Shading Language ES 3.00
2D array and 3D textures
Multisampled renderbuffers
Transform feedback
Uniform buffer Objects
Vertex Array Objects
Sampler Objects
Pixel Buffer Objects
Buffer-to-Buffer Copies
Boolean occlusion queries
Instanced rendering
Multiple render targets
Texture storage specification
R and RG textures
Seamless cube maps
Non-power of two textures
Texture LOD clamps

Mipmap level base offset and max clamp
At least 32 textures, at least 16 each for vertex/fragment shaders
16-bit and 32-bit floating-point textures
32-bit, 16-bit and 8-bit signed and Unsigned integer format renderbuffers, textures and vertex attributes
8-bit sRGB textures and framebuffers
11/11/10 floating-point RGB textures
Shared exponent RGB 9/9/9/5 textures
10/10/10/2 unsigned normalized and unnormalized integer textures
10/10/10/2 signed and unsigned normalized vertex attributes
16-bit floating-point vertex attributes
8-bit-per-component signed normalized textures
Sized internal texture formats
Three.js

Ricardo Cabello (Mr.doob)
What is WebVR?

Javascript API enabling VR experiences embedded in the traditional web.

Detect available Virtual Reality devices.

Query the device’s capabilities.

Poll the device’s position and orientation.

Display images at appropriate frame rate and with correct distortion.

Support from Google, Microsoft, Mozilla, Oculus, Samsung
WebVR on Chrome - Mobile

WebVR in Chrome for Android, including gamepad extensions

Available for Daydream now in Chrome 56

Released as an Origin Trial

Cardboard coming soon

WebVR “1.1” spec. Not final!
WebVR on Chrome - Desktop

WebVR in Chrome for Windows desktop, including gamepad extensions

Finalizing plans for inclusion in ToT Chrome/Chromium

Experimental builds available now at webvr.info
WebVR on other browsers

WebVR is available from several other vendors as well!

Firefox Nightly

Oculus Carmel

Samsung Internet Browser for Gear VR (WebVR 1.0)
Working on WebVR “2.0”

Updating spec to reduce platform incompatibility, clean up edge cases.

Enable API use in workers.

Support wider range of tracking needs. (Inside out)

Getting feedback from battle hardened spec veterans.

Avoiding antipatterns, interact better with the web platform.

Expect breaking changes before stable!
So what’s Khronos got to do with it?

WebVR is not a Khronos standard, but...
WebVR 💖 WebGL!

WebGL is currently the only way to render scenes for WebVR

“Path of least resistance” for getting pixels to the HMD

No need to retrofit VR onto a large number of Web APIs to get started

Techniques from outside the web (especially mobile VR) apply nicely

Make use of strong, existing ecosystem
WebVR 💖 (the idea of) OpenXR!

WebVR is only an interface to the native APIs

Browser has to choose which APIs (and devices) it cares about

Ideally implementations could largely rely on a common standard

With a standard API new devices could be available instantly

Working with OpenXR group to ensure web's needs are represented
Thanks!
Questions?

Brandon Jones - Google
Email: bajones@google.com
Twitter: @tojiro
Find out more at https://webvr.info
glTF Ecosystem Update
GDC, February 2017

Neil Trevett
Vice President Developer Ecosystem, NVIDIA | President, Khronos | glTF Chair
ntrevett@nvidia.com | @neilt3d
glTF - Runtime 3D Asset Delivery

New market opportunities for 3D content creation and deployment!

Audio
- MP3

Video
- H.264

Images
- JPEG

3D
- glTF

model/gltf+json MIME type Approved by IANA
Compact to Transmit  ✓
Fast to Load  ✓
Describes Full Scenes  ✓
Runtime Neutral  ✓
Extensible  ✓
Strong glTF Momentum

Oculus Executive Calls For 3D Equivalent Of JPEG To Build The Metaverse

A new standard for 3D scenes is gaining momentum with support from graphics industry leaders, potentially laying the groundwork for science fiction's "metaverse" to be realized.

The GL Transmission Format (gltF) from The Khronos Group, a computer graphics industry standards body, could also put magnitudes more 3D content on the Internet. The Khronos Group is responsible for a variety of technologies critical to today's virtual and augmented reality systems, and has been building out the ecosystem around g Multiscale Rendering (gMSR) to support a variety of use cases and deployment scenarios.

The glTF format is designed to be a lightweight, open, and easy-to-use solution for 3D models. It allows for efficient storage and transmission of 3D models, and supports a wide range of features such as textures, animations, and materials. The format is also designed to be backward-compatible with existing 3D formats, making it easier for developers to transition to glTF.

The glTF format has been gaining popularity in recent years, with support from major companies such as Apple, Adobe, and Google. The format is also being adopted by various industries, including architecture, engineering, and manufacturing, where 3D models are used for visualization and design.

In conclusion, the glTF format is a promising new standard for 3D models, with support from industry leaders and a growing ecosystem of tools and services. Its lightweight and efficient design makes it well-suited for a wide range of use cases, and its backward compatibility ensures a smooth transition for developers and users alike.
glTF Milestones

All glTF spec development on open GitHub: https://github.com/KhronosGroup/glTF

- Design Iteration and Multiple Implementations
- glTF 1.0 Spec Ratified and Released
- Validator Project
- glTF 2.0 Target Spec Finalization

Original motivation: standardized way to deliver 3D into WebGL applications

Significant Industry Adoption

glTF 2.0 adds Physically Based Rendering for higher-quality materials and rendering API independence

We are here! Seeking feedback on developer preview glTF 2.0 before finalization
Rapid Transition to glTF 2.0

• There are breaking changes from 1.0 to 2.0 - but processing is streamlined and simplified
  - Overwhelming community feedback to take the pain now - NOT significant work to upgrade

• Industry moving quickly to glTF 2.0 — lots of early adopters
  - BabylonJS, three.js, Cesium, xeogl, instant3Dhub

• gltf-pipeline includes glTF 2.0 updates — including glTF 1.0 to glTF 2.0 translator
  - Open source - use this to support both glTF 1.0 and 2.0 or move your users to 2.0

• Converters/Translators/Validators glTF 2.0 updates nearly ready
  - COLLADA2GLTF and obj2gltf translators
  - Khronos Validator and Gltf-test

• Samples and Tutorials
  - glTF 2.0 sample models with PBR are emerging
  - Extensive glTF tutorial series in draft

Consider moving your pipeline to glTF 2.0 ASAP
Blender glTF 2.0 Exporter RFQ

• Khronos-funded project to bring glTF 2.0 export to Blender
  - Blender, has some early work on glTF export: https://github.com/Kupoman/blendergltf
  - Project is to build out that work to glTF 2.0
  - Resultant code is to be contributed, royalty-free to the Blender open source project

• RFQ Milestones
  - 1. February 27th — Khronos Releases RFQ
  - 2. March 17th — RFQ responses received by Khronos
  - 3. March 24th — Contractor selected and notified
  - 4. March 29th — Contract executed and start of work

Details here:
https://www.khronos.org/rfq/
Please Consider Making a Bid!
glTF Roadmap Discussions

- **Mesh Compression**
  - Google Draco team

- **Progressive Geometry Streaming**
  - Fraunhofer SRC

- **Unified Compressed Texture Format for Transmission**
  - Basis format from Binomial
  - Optimized transmission format with efficient local expansion to any GPU format

- **Lighting Extension**
  - Enhanced lighting control

- **Extensions for API and language specifics**
  - Optional hooks for enhanced perf/functionality
  - Vulkan, DX12, Metal, GLSL, HLSL, SPIR-V, Metal C++

Share your roadmap priorities with us!

https://github.com/KhronosGroup/glTF
What’s new in glTF 2.0

• Physically Based Rendering (PBR) material definitions
  - Material information stored in textures
• Graphics API neutral
  - GLSL materials moved to extension
  - Proven by implementations using WebGL, Vulkan and Direct3D
• Improvements
  - Binary glTF in core
  - Enhanced Performance
Physically Based Rendering

- Standardize the BRDF inputs for common PBR workflows
  - Metallic-Roughness and Specular-Glossiness
- Incredible industry effort
  - Started by Fraunhofer and supported by Microsoft, Sketchfab, NVIDIA, Autodesk, Marmoset, University of Pennsylvania, and others

Sketchfab User: theblueturtle
https://sketchfab.com/models/b81008d513954189a063ff901f7abfe4
glTF 2.0 Physically Based Rendering

• In Core: Metallic-Roughness Material model
  - baseColor — base color
  - metallic — metalness
  - roughness — roughness

• Simple to implement with small resources
  - Can be everywhere

• Extension: Specular-Glossiness Material model
  - diffuse — reflected diffuse color
  - specular — specular color
  - glossiness — glossiness

• A little more resource heavy
  - Optional extension (e.g. on low-power devices)

• The two models can be combined
glTF 2.0 Scene Description Structure

- `.gltf (JSON)`
  Node hierarchy, PBR material textures, cameras

- `.bin`
  Geometry: vertices and indices
  Animation: key-frames
  Skins: inverse-bind matrices

- `.png`
  `.jpg`
  ... Textures

Geometry  
Texture based PBR materials
glTF 2.0 PBR materials in various engines

WebGL reference implementation
http://www.seas.upenn.edu/~moneimne/WebGL-PBR/

Laugh Engine running on Vulkan
https://github.com/jian-ru/laugh_engine
Calls to Action

• Evaluate/implement glTF 2.0
  - glTF 2.0 draft spec: https://github.com/KhronosGroup/glTF/tree/2.0/specification/2.0
  - glTF 2.0 Sample Models https://github.com/KhronosGroup/glTF-Sample-Models

• Bid on the Blender exporter RFQ!
  - Help support the glTF exporter ecosystem https://www.khronos.org/rfq/

• glTF Online Resources
  - Resource Hub https://www.khronos.org/gltf/
  - Github Page https://github.com/KhronosGroup/glTF

• Join Khronos!
  - Get directly involved in the glTF Working Group