Khronos Webinar

WebGL 2.0 is Here: What You Need To Know

2017 April 11

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

Slides: http://khr.io/webgl2webinar
Logistics

● Submit your questions
  ○ We’ll answer them at the end
  ○ Please include a slide number if needed

● Open the slides on your computer
  ○ If you want to try the demos without lag
  ○ http://khr.io/webgl2webinar
Background on WebGL

- **WebGL 1.0 was released in 2011**
  - Exposed 3D rendering to the Web Platform for the first time (GL ES 2.0)

- **Used for all kinds of 3D presentation on the web today**
  - Artists showcase their work on [Sketchfab](http://sketchfab.com), with over 1 million models shared and embedded across the web
  - The New York Times publishes visualizations in [Three.js](http://threejs.org)
    - Jupiter and Its Moons
    - The Dawn Wall, El Capitan’s Most Unwelcoming Route
  - Create amazing, fast, and interactive mathematical visualizations using [MathBox](http://mathbox.net)
Background on WebGL

● Also widely used in web games
  ○ Via cross-compilation (asm.js/WebAssembly)
    ■ Unity’s HTML5 target
      ● Publish Unity games straight to the Web with HTML5 and WebGL
  ○ And by pure-web 3D and game engines - mostly open-source
    ■ PlayCanvas, Turbulenz, BabylonJS, Three.js, …
    ■ WebGL-accelerated 2D game engines
  ○ For WebVR content
    ■ A-Frame, Three.js, …
Why target WebGL?

● Everyone has a web browser
  ○ No installation
  ○ WebGL (1.0) “just works” almost everywhere!
    ■ WebGL Stats - 96%
    ■ caniuse - 91%

● Many ways to using WebGL
  ○ Engines, libraries, etc.
And now...
WebGL 2.0 is here!

● WebGL 2.0 now available in Chrome/Firefox!
  ○ Firefox: Released on all platforms (January)
  ○ Chrome: Released to desktop platforms (January) - Android coming (late April)
  ○ Edge/Safari: Plan to ship WebGL 2.0
  ○ Desktop support already at 58% [WebGL Stats]

● WebGL 2.0.0 conformance status
  ○ Spec & test suite on track for ratification by Khronos
    ■ Very thorough - runs 10x longer than WebGL 1.0 tests
  ○ Chrome
    ■ Windows/macOS/Linux/Android: Passing 100% in Chrome 58
  ○ Firefox
    ■ Windows/macOS/Linux/Android pending; very nearly conformant
WebGL 2.0: Features at a Glance

- WebGL 2.0 exposes the OpenGL ES 3.0 feature set
  - Brings desktop and mobile platforms’ features much closer together - better portability
- Many WebGL 1.0 extensions now core
- Features
  - 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 1.0
WebGL 2.0: Features at a Glance (continued)

- Features
  - 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*
Teasers
WebGL 2.0 + WebVR

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

Procedural clouds
Procedural water ripples and reflection
Leaf particle system
Animated trees
Mirrored surfaces
Dynamic lights
Online asset streaming
Runtime lightmap baking
& more
Feature Tour

● With more demos!
  ○ (And some info on the techniques)

● Featuring:
  ○ **WebGL 2 Samples Pack**
    ■ Short and easy to understand code samples demonstrating WebGL 2.0 features
    ■ Trung Le & Shuai Shao, Patrick Cozzi
  ○ **WebGL 2 Examples**
    ■ Standalone rendering algorithm demos implemented in raw WebGL 2.0
    ■ Tarek Sherif
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
Textures: More Formats

- sRGB formats
  - Textures for linear rendering techniques [GPU Gems; non-web] (pictured)
  - Gamma correction applied before sampling & filtering - more correct
  - Essential for high dynamic range rendering & physically based rendering
  - 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 rendering techniques\(^1\) & simulation (fire, smoke, fog, etc.)
  - Medical imagery\(^1\)
  - Cached lighting effects
  - Procedural textures\(^1\)
    - 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

\(^1\) GPU Gems chapters
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 directly
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] (pictured)
  ○ 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
  - Multiple fragment outputs from one fragment shader: gl_FragData[...]
- Critical for deferred shading (and related techniques)
  - Render the entire scene just once, into a series of intermediate textures
  - Efficiently shade with many dynamic lights - shade the intermediate textures once per light

[images due to Astrofra via Wikipedia]
Multiple Render Targets

● Big performance gains for many types of scenes (and hardware)
  ○ N+M compute instead of N*M… but higher memory load!

● Demos
  ○ webgl2examples: Deferred Rendering [Tarek Sherif]
  ○ WebGL Deferred Shading [Sijie Tian, etc.] (pictured)
  ○ Deferred Irradiance Volumes [Florian Bösch]
  ○ Deferred Rendering [m_panknin]
  ○ Spiral waves in excitable media [Evgeny Demidov]
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, birth/death, steps physics each frame
  - After the Flood: particle system leaves
  - Lorenz strange attractor [Evgeny Demidov] (pictured)
  - Samples pack, webgl2examples
- Non-web OpenGL examples:
  - OpenGL 4.0 review [Christophe Riccio], Noise-Based Particles advection shader [Philip Rideout]
Integer vertex data

- Go hand-in-hand with transform feedback, as well as integer texture support
  - E.g. maintain a pseudo-random number generator’s state per-vertex

- More generally:
  - Can send multiple integers into the vertex shader
  - Perform logical and arithmetic operations on them
  - Output them (to fragment shader or transform feedback)
Multisampled renderbuffers

● Efficient antialiased offscreen rendering
  ○ User controls number of samples dynamically
  ○ Useful for implementing techniques like vector curve rendering
    ■ Three.js vector curve rendering (pictured), Rendering SVG paths in WebGL
  ○ Advanced operations on multisample data
  ○ 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 volume rendering algorithms [GPU Gems]
- With 3D textures and 2D texture arrays, 3D GPU fluid simulations
More features for performance optimization

- **Uniform buffers (potentially significant speedups)**
  - Allows updating many uniforms in a batch with one API call
  - samples pack, webgl2examples

- **Vertex array objects (also a WebGL 1.0 extension)**
  - Encapsulates all state that is associated with the vertex processor
    - Holds references to the vertex buffers, index buffer, and layout specification
    - Usually set up once - much quicker to switch VAO than all vertex state

- **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 (pictured)
  - 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 (pictured)
  - Centroid interpolation
  - Fragment discard
  - Flat/smooth interpolators
  - Non-square matrices
  - Sampling mipmapped textures by LOD
  - ...

Slides: [http://khr.io/webgl2webinar](http://khr.io/webgl2webinar)
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
PlayCanvas: *After the Flood*

- **Techniques used**
  - HDR+MSAA rendering (floating point + multisampling)
  - Using z-buffer as a texture (depth textures)
  - Procedural clouds (3D textures)
  - Hardware *percentage-closer* filtering for shadow maps
  - *Alpha to coverage*
  - Particle system leaves (transform feedback)
WebGL 2 Examples - Tarek Sherif

- [https://github.com/tsherif/webgl2examples](https://github.com/tsherif/webgl2examples)
  - Rendering algorithms implemented in raw WebGL 2
  - Order-Independent Transparency & Screen Space Ambient Occlusion
    - Vertex Arrays, Uniform Buffers, Multiple Render Targets, Float Textures, texelFetch, Instancing
  - Depth of Field (pictured)
    - Vertex Arrays, Uniform Buffers, Depth Textures, texelFetch, Instancing
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 Web 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 capabilities
  - 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
  - Brandon Jones, Victor Miura, & many more
- 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
  - Trung Le & Shuai Shao, Patrick Cozzi; University of Pennsylvania
- PlayCanvas, Sketchfab, Floored, Google Maps
- The Three.js community
  - esp. Ricardo Cabello
- ...and many more community collaborators
  - Authors of linked demos
    - Tarek Sherif, BioDigital
    - 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
Questions?

- WebGL spec, GitHub, mailing list, etc.
  - [webgl.org](http://webgl.org)
- WebGL 2 Samples Pack
  - [github.com/WebGLSamples/WebGL2Samples](https://github.com/WebGLSamples/WebGL2Samples)
- Tarek Sherif’s WebGL 2 Examples
  - [github.com/tsherif/webgl2examples](https://github.com/tsherif/webgl2examples)
- Ken Russell and Zhenyao Mo also here for questions