The state of WebGL and glTF
Patrick Cozzi | @pjcozzi
3D Formats Working Group
September 2015
Khronos Connects Software to Silicon

Open Consortium creating ROYALTY-FREE, OPEN STANDARD APIs for hardware acceleration

Defining the roadmap for low-level silicon interfaces needed on every platform

Graphics, compute and vision processing

Rigorous specifications AND conformance tests for cross-vendor portability

Acceleration APIs BY the Industry FOR the Industry

Well over a BILLION people use Khronos APIs Every Day...
Today’s Topics

- What’s new? What’s upcoming?
  - Spec and ecosystem

WebGL™  glTF™

Today’s Main Focus
A Few Recent WebGL Apps

**Marmoset Viewer**
- http://www.marmoset.co/viewer/gallery
- PBR materials and IBL

**Red Bull X-Alps**
- Paragliding and hiking adventure race. 1,000 km across the Alps
- Collaboration between Red Bull X-Alps, Doarama (https://www.doarama.com), and Cesium (http://cesiumjs.org/)

**NASA Curiosity Rover: Blend4Web**
- Blend4Web allows artists and designers to develop sophisticated interactive content without extensive coding
- App: http://eyes.nasa.gov/curiosity/
Strong WebGL Ecosystem

- **Platforms**
  - Desktop, Android, and iOS

- **Engines**
  - Three.js, BabylonJS, SceneJS, xeoEngine, Pex, Blend4Web, Cesium

- **Tools**
  - Browser debuggers/profilers, Shader Editors
    - WebGL Inspector, Web Tracing Framework

- **Education**
  - Books, online courses, and tutorials

- **Adoption across many domains, not just games**

For tools, see: http://www.realrendering.com/blog/webgl-debugging-and-profiling-tools/

xeoEngine - Data-driven object-component-based 3D engine on WebGL
Pex - http://vorg.github.io/pex/

Chrome shader editor extension: https://github.com/spite/ShaderEditorExtension

Beyond games: geospatial, medical visualization, ...
WebGL 2.0

- Brings most OpenGL ES 3.0 features to WebGL to enable
  - Better visual quality
  - Better performance
  - New GPU algorithms
- Draft spec is available
  - https://www.khronos.org/registry/webgl/specs/latest/2.0/
- Prototype implementations in Chrome and Firefox
  - https://www.khronos.org/webgl/wiki/Getting_a_WebGL_Implementation
- Chromium aiming to pass all known conformance tests by the end of the year
- Many WebGL 2.0 features are available today as extensions
  - ANGLE_instanced_arrays, OES_vertex_array_object, WEBGL_draw_buffers, ...

The tests are all in https://github.com/KhronosGroup/WebGL under sdk/tests/conformance2/ and sdk/tests/deqp.
Major WebGL 2.0 Features

- WEBGL_draw_buffers (Multiple Render Targets)
  - Deferred shading, Order-Independent Transparency (OIT), ...

Left image: https://hacks.mozilla.org/2014/01/webgl-deferred-shading/

Right image: http://cesiumjs.org/2014/03/14/Weighted-Blended-Order-Independent-Transparency/
Major WebGL 2.0 Features

- Instancing
  - Less draw calls to draw lots of similar objects (instances)

Image from https://cesiumjs.org/2015/01/05/Cesium-version-1.5-released/
Major WebGL 2.0 Features

- Uniform buffers
- Transform feedback
- Multisampled Renderbuffers
- 3D textures
- NPOT textures
- More texture formats
- Occlusion queries
- Vertex array objects
- Sampler objects
- Sync objects
- Fragment depth
- Primitive restart

For WebGL 1 and 2 stats for your browser, check out http://webglreport.com/
Beyond graphics APIs, we need a runtime 3D Model Format
What’s in a 3D model?

- Node hierarchy and geometry
- Materials and textures
- Animations and skins

Not shown: lights and cameras

Get data in or out of modeling tools without losing any information. Like a .bmp file compared to a .jpg (for runtime).

Sometimes there is an SDK to read/write these files.

Sometimes the format is published.

.ma – ASCII
.mb – binary

Blender logo: http://www.blender.org/about/logo/
Maya logo: http://en.wikipedia.org/wiki/Autodesk_Maya
Modo logo: http://en.wikipedia.org/wiki/Modo_(software)
COLLADA is an open standard from Khronos. Open-source OpenCOLLADA and COLLADA DOM read/write COLLADA file.

FBX is proprietary and owned by Autodesk. Autodesk has an SDK to read/write FBX. FBX can be binary or ASCII. There is an unofficial spec.

OBJ is originally from Wavefront. It is geometry only, so doesn’t include animations, skins, physics, etc.
Interchange formats can move assets between tools, but what about between tools and the runtime engine?
Interchange Formats

- Target tools - not the graphics API
- Example: COLLADA
  - XML + image files
  - One index per attribute, not vertex
  - Unsigned int indices
  - Transform stack per node
  - Polygons and splines
  - Doesn’t specify image file format
  - Lots of flexibility and indirection in animations and skins

Interchange formats are generally verbose and slow to load for runtime use. Interchange formats need to go through many conversion steps before a graphics API. This doesn’t belong in a runtime; it belongs in the content pipeline.

Keyframe animation supports several different splines. Great for interchange, but a runtime usually only needs one or two.
The content pipeline runs offline, perhaps as part of the build process. It does not ship with the engine.

However, generally, each engine has its own proprietary runtime format and its own content pipeline!
3D Needs a Transmission Format!

- Bridge the gap between tools and ‘GL’ based apps
  - Reduce duplicated effort in content pipelines
  - Enable richer 3D representation - OBJ, STL etc. too limited
  - Provide common publishing format for content tools and services

<table>
<thead>
<tr>
<th>Audio</th>
<th>Video</th>
<th>Images</th>
<th>3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP3</td>
<td>H.264</td>
<td>JPEG</td>
<td>?</td>
</tr>
<tr>
<td>napster</td>
<td>YouTube</td>
<td>Facebook</td>
<td>![ ]</td>
</tr>
</tbody>
</table>

A widely adopted format ignites previously unimagined opportunities for a media type
glTF = “JPEG for 3D”

- ‘GL Transmission Format’
  - Runtime asset format for WebGL, OpenGL ES, and OpenGL applications
- Compact representation for download efficiency
  - Binary mesh and animation data
- Loads quickly into memory
  - GL native data types require no additional parsing
- Full-featured
  - 3D constructs (node hierarchy, materials, animation, cameras, lights)
- Runtime Neutral
  - Can be created and used by any tool, app, or runtime
- Flexible Extensibility
  - E.g. payloads with compression and streaming
glTF Internals

- JSON: cross-platform, compact, readable, allows validation, and minifies and compresses well
- Geometry, animation, and skins are binary, unlike COLLADA, for example, which uses XML
- Binary data is little endian
- Binary blobs - allow efficient creation of GL buffers and textures since they require no additional parsing (except perhaps decompression with a glTF extension)
- Shaders can be in .json or separate .glsl files
- Can have any number of .bin files
- Flexible for a wide array of applications.
Bottom-up:

Geometry
- buffer – binary blob. Can be combination of geometry, animation, and skins
- bufferView – subset of buffer with target info (ARRAY_BUFFER, ELEMENT_BUFFER, animation/skin)
- accessor – subset of bufferView with type info, e.g., float-point. Similar to a call to glVertexAttribPointer
  - For example, a bufferView may be all vertices in the asset (think glBufferSubData), where as an accessor may be an individual attribute for a mesh (think glVertexAttribPointer)
- mesh – (composed of primitives, not shown) – corresponds to glDrawElements and glDrawArrays
- node – one or more meshes, plus transform, plus children, plus optional skin.

Material
- image – Image file
- sampler – texture filter and wrap modes, think glTexParameter
- texture – think glTexImage2D
- shader – GLSL shader source
- program – think glCompileShader and glLinkProgram
- technique – parameter inputs (attributes + uniforms) + pass – program + render state
- Material – an instance of a technique. Overrides parameter inputs

Animation
- animation accesses keyframes from accessor
- animation targets node [transforms], material/technique parameters, and camera/light

Skin
- skin accesses inverse-bind matrices from accessor
- node references skins. skins reference nodes
**glTF Project Status**

- **Open specification; Open process**
  - Specification and multiple loaders and translators in open source
  - [https://github.com/KhronosGroup/gltf](https://github.com/KhronosGroup/gltf)

- **Draft glTF 1.0 spec is imminent**
  - spec-1.0 branch

- **Extension mechanisms fully defined**
  - Vendor, multi-vendor and official Khronos extensions (mirrors OpenGL)
  - Anyone can ship vendor extensions at any time - no permissions needed
  - First extensions will be included in launch
Babylon team had initial implementation (without skinning) in less than a week!
Initial glTF Extensions

- **KHR_binary_gltF (Khronos extension)**
  - Enables a glTF file to refer to external binary asset packages

- **EXT_quantized_attributes (vendor extension)**
  - Quantization-based attribute compression
  - Decompression in vertex shader

- **Open3DGC - MPEG 3D mesh compression (in progress)**
  - C++ encoder/decoder + JavaScript decoder
  - 40-80% compression
  - Extensions inserts decompression between file buffer and vertex data
  - Building support into COLLADA2GLTF converter and Cesium loader

EXT_quantized_attributes - https://github.com/KhronosGroup/glTF/pull/414

Open3DGC - https://github.com/KhronosGroup/glTF/issues/398
## Open3DGC glTF Extension Initial Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Vertices</th>
<th>Tris</th>
<th>Flat + Gzip</th>
<th>Open3DGC + Gzip</th>
<th>Compression Amount</th>
<th>JavaScript Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLADA Duck</td>
<td>2.1k</td>
<td>4.2k</td>
<td>54 KiB</td>
<td>14 KiB</td>
<td>-74%</td>
<td>24 ms</td>
</tr>
<tr>
<td>Stanford Bunny</td>
<td>2.5k</td>
<td>5.0k</td>
<td>105 KiB</td>
<td>56 KiB</td>
<td>-47%</td>
<td>30 ms</td>
</tr>
<tr>
<td>Stanford Dragon</td>
<td>435k</td>
<td>871k</td>
<td>7792 KiB</td>
<td>2141 KiB</td>
<td>-73%</td>
<td>630 ms</td>
</tr>
<tr>
<td>3D Tile</td>
<td>12.8k</td>
<td>6.5k</td>
<td>102 KiB</td>
<td>59 KiB</td>
<td>-42%</td>
<td>–</td>
</tr>
<tr>
<td>OpenStreetMap NYC</td>
<td>—</td>
<td>—</td>
<td>337 MiB</td>
<td>207 MiB</td>
<td>-39%</td>
<td>(Streamed)</td>
</tr>
</tbody>
</table>

Google Chrome 44.0, Windows 8.1, Intel i7-4980HQ @ 2.80GHz

Decompression can be done in a web worker.
Cesium 3D Tiles Using glTF

- An open specification for streaming massive 3D geospatial datasets
  - Streams 3D content including buildings, trees, point clouds, and vector data
- Hierarchical Level of Detail (HLOD)
  - Only visible and prioritized tiles are streamed
  - glTF payloads can be compressed, e.g., using 3DGC extension

Demo: cesiumjs.org/NewYork
Get Involved!

- **WebGL**
  - Experiment with WebGL 2 implementations:
    [https://www.khronos.org/webgl/wiki/Getting_a_WebGL_Implementation](https://www.khronos.org/webgl/wiki/Getting_a_WebGL_Implementation)

- **glTF**
  - Review the spec:
    [https://github.com/KhronosGroup/glTF/blob/spec-1.0/specification/README.md](https://github.com/KhronosGroup/glTF/blob/spec-1.0/specification/README.md)

- **Contact**
  - Patrick Cozzi
  - @pjcozzi
  - pjcozzi@siggraph.org