glTF Update

Tony Parisi, Patrick Cozzi, Kai Ninomiya
3D Formats Working Group

August 2015
3D Needs a Transmission Format!

- Need to bridge the gap between tools and today’s 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

- Why is 3D the last data type with an agreed transmission format?

<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>

An effective and widely adopted codec 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 fast download
  - Meshes, skins and animation data etc. binary files/typed arrays
  - Extension capability for future formats with compression and streaming

- Loads quickly into memory
  - JSON for scene structure and other high-level constructs
  - GL native data types require no additional parsing

- Full-featured and pragmatic
  - 3D constructs (hierarchy, cameras, lights, common materials, animation)
  - Full support for shaders and arbitrary materials

- Runtime Neutral
  - Can be created and used by any tool, app or runtime
Some JSON

Describing scene structure

```
"nodes": {
  "LOD3sp": {
    "children": [],
    "matrix": [],
    "meshes": [
      "LOD3spShape-lib",
      "name": "LOD3sp"
    ],
  },
}
```

Defining a mesh

```
"meshes": {
  "LOD3spShape-lib": {
    "name": "LOD3spShape",
    "primitives": [
      {
        "attributes": {
          "NORMAL": "accessor_25",
          "POSITION": "accessor_23",
          "TEXCOORD_0": "accessor_27"
        },
        "indices": "accessor_21",
        "material": "blinn3-fx",
        "primitive": 4
      }
    ],
  }
},
```

Referencing buffers

```
"bufferViews": {
  "bufferView_29": {
    "buffer": "duck",
    "byteLength": 25272,
    "byteOffset": 0,
    "target": 34963
  },
  "bufferView_30": {
    "buffer": "duck",
    "byteLength": 76768,
    "byteOffset": 25272,
    "target": 34962
  }
},
```

Describing scene structure

Defining a mesh

Referencing buffers
Project Status

• Open specification; Open process
  - Specification and sample code: https://github.com/KhronosGroup/glTF
  - Multiple implementations in sample source

• glTF 0.8 schema available
  - Getting very close to glTF 1.0 - most likely no major breaking changes in 1.0

• Features TBD
  - Extensions e.g. Mesh Compression
  - Cube maps

• Next steps
  - Draft 1.0 target date September 23 (Graphical Web conference)

We’re looking for your feedback!
glTF Adoption

three.js loader
(updates coming 9/15)
https://github.com/mrdoob/three.js/

BabylonJS
(under development)
http://www.babylonjs.com/

Cesium - it's the native format!
http://cesiumjs.org/

PIPELINE TOOLS

collada2gltf converter
https://github.com/KhronosGroup/glTF

Online drag and drop converter
http://cesiumjs.org/convertmodel.html

FBX to glTF
(under development)
http://gltf.autodesk.io/
glTF Extensibility

• glTF
  - Simple format
  - Need more?
    - Extras and extensions on any object

• Extras
  - For app-specific data
    - mesh.extras.description: { ... }

• Extensions
  - For new general-purpose functionality specs
    - bufferView.extensions.mesh_compression_open3dgc: { ... }
Open3DGC Mesh Compression

- Open3DGC mesh compression library (Khaled Mamou, AMD; MIT-licensed)
  - 40-80% compression over flat arrays
    - Fast decompression
  - C++ encoder/decoder + JavaScript decoder
    - Floating-point quantization, parallelogram prediction, animations, etc.


- In glTF
  - Insert decompression between file buffer and vertex data

- WIP encoder support in COLLADA2GLTF
  - Static models
  - Some support for uncompressed animation data

- WIP decoder support in Cesium
  - Very straightforward: about 1 workday (static models)
  - Supports decompression in a Web Worker

- Feedback welcome
  - Join the discussion on GitHub!
    - https://github.com/KhronosGroup/gltF/issues/398
Sample Results

- Comparison of
  - Default flat-array mesh encoding + gzip
  - Open3DGC, ASCII-mode + gzip
    - Compression parameters tuned manually for quality

<table>
<thead>
<tr>
<th>Model</th>
<th>Verts</th>
<th>Tris</th>
<th>Flat+Gzip</th>
<th>O3DGC+Gzip</th>
<th>JavaScript</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>
</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>
</tr>
<tr>
<td>Stanford Dragon</td>
<td>435k</td>
<td>871k</td>
<td>7792 KiB</td>
<td>2141 KiB</td>
<td>-73%</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>
</tr>
<tr>
<td>OpenStreetMap NYC</td>
<td>—</td>
<td>—</td>
<td>337 MiB</td>
<td>207 MiB</td>
<td>-39%</td>
</tr>
</tbody>
</table>

- Σ: Dragon decompressed in 7 parts (64k vertices each)
- Google Chrome 44.0, Windows 8.1, Intel i7-4980HQ @ 2.80GHz
Demos