The runtime asset format for GL-based applications

Specification Editors
Patrick Cozzi | AGI
Tony Parisi | Wevr
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 describes node hierarchy
  - Includes cameras
  - References geometry, animations, skins, shaders, textures
- Vertices
  - Uses native typed array format
  - Includes key-frame animations and skinning
- Shaders
  - With extensions for materials
- Textures
  - Use existing standard image compression formats e.g. JPEG
- Extras
  - For app-specific data (metadata)
Three.js

• The most popular WebGL library
  - Github repo: 22k favorites; 7,200 forks; 1,500 watchers
  - Mr.doob 31k Twitter followers
• V1.0 importer integrated into dev branch 8-Mar-2016 with handful of samples
• Importer code also included in main repo with more samples

model from 3drt.com
Request for Quotations (RFQ) to create glTF Validator is out!

The glTF Ecosystem tools, translators, and engines are used to work with glTF files. The tools include Blender DIRECT export, Autodesk FBX -> glTF, COLLADA2GLTF, Cesium converter, OBJ2GLTF, and glTF Pipeline. The translators convert and optimize glTF files, while the engines, such as PEX and CESIUM, import glTF files. The validator validates the exported glTF files. The glTF Ecosystem Page is available at https://github.com/KhronosGroup/gltf#gltf-tools.
glTF Ecosystem

Learn
Sample models
Open source code
  JavaScript, C#, C++
Readable spec

Extensions and Derived Formats
Binary glTF
SRC
3D Tiles

https://github.com/KhronosGroup/glTF
BabylonJS

• glTF loader with animation and skins
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

Over 1.1 million OpenStreetMap buildings in New York City
Cesium 3D Tiles Using glTF

Open Source
3D City Database

virtualcitySYSTEMS

virtualcityMAP

fully automated process chain

CityGML WFS
WFS / WMS / ...

queries to the 3D city model
integration of 2D/3D geodata

2D/3D geo data

OGC services
Pex

- Initial glTF loader (geometry and materials)
- [https://github.com/pex-gl/pex-gltf](https://github.com/pex-gl/pex-gltf)
xeoEngine

- glTF is native format (geometry and materials)
- Tutorial: https://github.com/xeolabs/xeoengine/wiki/Importing-glTF
glTF Validator RFQ! Consider Bidding!!

- Validate glTF assets: geometry, material, animation, and skins
  - Ensure compliance with spec and schema

- Augment glTF sample models to cover corner cases

- Optional features
  - Binary glTF, REST service, drag-and-drop validator, drag-and-drop three.js viewer

- Schedule
  - March 15 - Khronos Released RFQ
  - March 31 - RFQ responses received by Khronos
  - April 15 - Contractor selected and notified
  - April 29 - Contract executed and start of work

- https://www.khronos.org/rfq
Upcoming PBR Extension

- Will extract from new WebGL 2.0 and ES2015 engine by Fraunhofer
- PBR Shading Model based on Unreal Engine & Marmoset Toolbag

- Albedo color or map
- Metallic factor or map
- Roughness factor or map
- Normal map
Please Get Involved!

- Consider bidding for Validator RFQ
- Create and use glTF exporters and tools
- Generate tutorials on your experience with glTF
- Make glTF examples for Vulkan, OpenGL, and OpenGL ES
- Submit glTF extensions to meet your business needs

https://github.com/KhronosGroup/gltf
Background
What’s in a 3D Asset or Model?

- Scene hierarchy and geometry
- Materials and textures
- Animations and skins
- Final Asset in Scene
glTF Internals
glTF Example

JSON Node (the truck) with three children (sets of two wheels)

Visualization of Node Hierarchy

Three animations - one for each set of wheels
3D Model Creation and Deployment - Today

Custom Content Pipeline

Authoring formats tend to be large and require lots of processing to parse
Transmit over network

Runtime App

Application has to be customized to understand custom formats - cannot accept assets from diverse servers
-> Silo'd content

>30 3D formats in use
OBJ/STL contain single-models NOT scenes
Need lights, cameras, animations, scene hierarchy etc.
3D Model Creation and Deployment Standards!

- **.blend**
- **.ma / .mb**
- **Collada**
  - Flexibly mix and match tools through authoring interchange format that understand full scenes
  - COLLADA is NOT a transmission format
    - Large 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
    - ....

- **Format Conditioner**
  - Can convert to glTF from any format

- **glTF transmission format carries full scenes: compact and easy to parse**

- **Runtime App**
  - Application can process received standard format 3D assets from any server
  - -> open and interoperable AR

- **Open COLLADA Importer/Exporter and COLLADA Conformance Tests on GitHub**

- **COLLADA2GLTF Translator on GitHub**

- **Three.js glTF Importer on GitHub**
glTF Project Status

- Open specification; Open process
  - Specification and multiple loaders and translators in open source
  - [https://github.com/KhronosGroup/glTF/blob/spec-1.0/specification/README.md](https://github.com/KhronosGroup/glTF/blob/spec-1.0/specification/README.md)

- glTF 1.0 spec finalized
  - Launched in October 2015!

- 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 included in launch

- More details
  - [https://www.khronos.org/gltf/](https://www.khronos.org/gltf/)
Launch Industry Support

“It was obvious for the babylon.js team that glTF was a must have feature in order to integrate well within the 3D ecosystem.”
David Catuhe, principal program manager at Microsoft and author of babylon.js

“glTF has some remarkable features that will make it simple for developers to include and run 3D digital assets in their web or mobile applications”
Cyrille Fauvel, senior ADN Sparks manager at Autodesk

“Unlocking 3D content from proprietary desktop applications to the cloud creates massive new opportunities for collaboration. This future is so close we can feel it - the hardware is capable, the browsers are capable, now if only we could solve the content pipeline. Go glTF!”
Ross McKegney, Platform @ Box

“Defining a 3D graphics transmission model is challenging due to the extensive diversity of 3D graphics representations and use cases and the 3D ecosystem is being held back by a lack of a simple and universally efficient data representation. glTF has an important role by defining a foundation on which application specific compression and transmission components can be incrementally added. We are looking forward to glTF extensions to enable efficient MPEG compression technologies for 3D graphics to be widely deployed.”
Marius Preda of the MPEG Consortium
glTF Adoption

three.js Loader
https://github.com/mrdoob/three.js/

It's the native format!
http://cesiumjs.org/

Microsoft

Babylon.js Loader (in development)
http://www.babylonjs.com/

PIPELINE TOOLS

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

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

FBX to glTF Convertor
(in development)
Drag and drop convertor coming
http://gltf.autodesk.io/

Native import and display of glTF models

3D Advertising Solutions with native glTF import
Initial glTF Extensions

- Any company can define glTF vendor extensions
  - Khronos manages extension name space
  - Popular extensions can be proposed to be adopted into standard extensions and then possibly into core

- KHR_binary_glTF (Khronos extension)
  - Enables a glTF file to use binary asset packages

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

- MPEG 3D mesh compression (in progress)
  - MPEG-SC3DMC codec (Scalable Complexity 3D Mesh Compression)
  - Uses Open3DGC open source - C++ encoder/decoder + JavaScript decoder
  - 40-80% compression for many 3D assets
  - Extensions inserts decompression between file buffer and vertex data
  - Building support into the COLLADA2GLTF converter and Cesium loader
# 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
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
            }
        ],
        "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
    }
}
```
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: { ... }`
Shaders -> Material Descriptions

- Physically Based Rendering (PBR) is hot topic in 3D graphics
  - Describe the properties of each material - not a literal shader
  - Easy for content creators
  - Realism of materials can scale to the platform capabilities

- Reach out to existing projects to include in transmission format?

NVIDIA MDL (material description language)

Open Shading Language


http://www.openshading.com/
COLLADA and glTF Ecosystem

- OpenCOLLADA Importer/Exporter and COLLADA Conformance Tests On GitHub
- Tool Interop
- COLLADA2GLTF Translator
- Other authoring formats
- Web-based Tools
  - Autodesk Maya
  - blender
- Pervasive WebGL deployment
  - Three.js glTF Importer
  - Rest3D initiative
- WebGL
Get Involved with glTF!

- glTF specification
  - Review and use the specification: https://github.com/KhronosGroup/glTF/blob/spec-1.0/specification/README.md

- More details
  - https://www.khronos.org/gltf/

- Questions and supportive quotes
  - ntrevett@nvidia.com
  - @neiltd3d
  - #gltf