3D Transmission Format

Neil Trevett
Khronos President
Vice President Mobile Content, NVIDIA
May 2013
3D Needs a Transmission Format!

- Compression and streaming of 3D assets becoming essential
  - Mobile and connected devices need access to increasingly large asset databases

- 3D is the last media type to define a compressed format
  - 3D is more complex - diverse asset types and use cases

- Need to handle sophisticated 3D assets and scenes
  - Scenegraph, geometry, textures, materials, animations...

<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
COLLADA = XML Interchange of 3D Assets

- COLLADA is a XML database schema for 3D assets
  - Can hold *everything* to do with a scene: geometry with full skinning, advanced material and visual effects, animation, physical properties and collisions

- COLLADA can be used to transport 3D assets between *tools*
  - Enables binding of diverse DCC and 3D processing tools into a production pipeline

- COLLADA is an open, archive-grade format that retains meta information
  - Retains all information - even multiple versions of the same asset

- COLLADA is NOT an transport format
  - Conditioning pipelines optimize the asset database for a target device or usage
COLLADA Conditioners Directory

- Encourage community to share COLLADA conditioners
OpenCOLLADA Import/Export

- OpenCOLLADA upgraded and released as open source on Github
  - High-quality, full functionality COLLADA importer/exporter
  - Available free-of-charge for tools vendors to integrate
  - [https://github.com/KhronosGroup/OpenCOLLADA](https://github.com/KhronosGroup/OpenCOLLADA)

- COLLADA conformance test suite released on GitHub
  - Enables tools vendors, and their users, to test for conformant COLLADA IO
  - [https://github.com/KhronosGroup/COLLADA-CTS](https://github.com/KhronosGroup/COLLADA-CTS)

No excuses left for missing or low quality COLLADA import/export! 😊
Significant 3D Ecosystem Announcements

Connecting the two with a new 3D run-time asset transmission format - glTF. A standards-based content pipeline for rich native and Web 3D applications.

Creating the Industry’s most portable and reliable 3D asset interchange format

Creating the Industry’s most portable and secure 3D execution platform
glTF - OpenGL Transmission Format

- **glTF** is a final-stage asset format driven by OpenGL APIs
  - Bridges the gap between COLLADA and GL APIs (OpenGL, OpenGL ES and WebGL)
  - COLLADA working group driving the glTF specification design

- Uses JSON to describe node hierarchy
  - Platform neutral, run-time neutral, many processing libraries available

- Node hierarchy refers to geometry, textures, materials, animations...
  - Separate binary blob for geometry/texture assets
  - Will enable parallel work on 3D asset streaming and compression

**DOES NOT IMPLY OR MANDATE ANY RUN-TIME BEHAVIOR**

- Draft spec available
  [https://github.com/KhronosGroup/glTF](https://github.com/KhronosGroup/glTF)
Building glTF

• Prototype implementation: COLLADA2GLTF
  - Uses OpenCOLLADA to convert COLLADA assets to JSON for use in WebGL
  - Source uploaded on Github: https://github.com/KhronosGroup/collada2json#readme

• Format requirements driven by WebGL/OpenGL ES
  - Unique indices buffer, requires shaders, split meshes > 65536 indices...
  - Buffers description compliant with typed arrays
  - Incorporate requirements from MPEG, Web3D and others
Building a Complete Transmission Solution

- Three components need for complete solution
  - JSON SceneGraph encoding +
  - REST3D APIs for Server/Client Negotiation +
  - 3D Asset payload compression/streaming

1. Send JSON encoded scene graph
2. Use REST to negotiate asset selection, compression and streaming
3. Compress and Stream Assets
Rest APIs for 3D Asset Serving

• RESTful APIs - enables a web services approach
  - Eliminates need to bake all use cases into complex format
  - Formats and capabilities can be incrementally rolled out

• Streaming options queried and negotiated through REST e.g.
  - Request geometry in viewing frustum first - then the complete data set
  - Download geometry first then textures
  - Request highly compressed geometry (lossy) then lossless geometry
  - Request specific payload compression that app heuristically knows works well for its domain

• Rest3D Initiative - informal discussion group - chaired by Remi Arnaud
  - http://rest3d.wordpress.com/
  - How do we work with Rest3D?
  - Do we share goals and timeline requirements?
Asset Compression/Streaming

- Need range of geometry compression options - Lossy vs. Lossless
  - Gzip is baseline for compression efficiency
- Need to identify and select compression technologies
  - MPEG has open-source and royalty-free mesh compression algorithms
    - 3D Mesh Coding Progressive Streaming (3DMC), Bones Based Animation (BBA)
  - X3D Binary Compression
  - WebGL-Loader used by Google body
  - X3DOM: Efficient binary meshes
  - OCG 3D Portrayal
    - Using lossless png for geometry and REST APIs for request section of city model
- Open call to the industry for payload compression techniques
  - Then select which ones should be referenced
  - Technical and licensing factors to be considered
Transmission Format Liaison

- Khronos suggested liaison working group for joint specification authoring
  - MPEG feedback was divide work into separable, individual projects at each SDO
  - Projects normatively reference projects from other SDOs

- glTF drafts publicly available for feedback and requirements gathering
- COLLADA2JSON open source example convertor

- glTF can refer to mesh and animation compression technology from MPEG
- Creates MPEG glTF loader

- REST APIs for web services
- Additional compression and streaming technologies

+ Other SDOs and companies

- Create X3D/X3DOM glTF loaders
Discussion

• Getting positive feedback to glTF so far from WebGL community
  - JSON description is not rocket science - it’s a simple substrate
  - Everyone was doing their own version anyway - saving everyone’s time
  - Even this simple step giving good loading speed-ups

• Is glTF a good substrate for the other SDOs?
  - Very open to input and feedback

• How do we drive definition of Rest APIs
  - Work with Rest3D or create different initiative? Where?

• How do we propose/select compression technologies?
  - Need royalty-free ratification at an SDO to be candidate
  - Industry-wide call for proposals?
  - What is the selection process?