glTF 2.0 Launch
Web3D Conference, June 2017

Neil Trevett | Khronos President
NVIDIA Vice President Developer Ecosystem
ntrevett@nvidia.com | @neilt3d
glTF - Efficient Runtime 3D Asset Delivery

Audio
MP3

Video
H.264

Images
JPEG

3D

New market opportunities for 3D content creation and deployment!

model/gltf+json MIME type Approved by IANA

Portable 3D Graphics Transmission Format
A vital building block for the Metaverse
Widely shared avatars, objects, information
Avoids siloed, per-app, per-platform content formats
Compact to Transmit
Fast to Load
Describes Full Scenes
Runtime Neutral
Open and Extensible
glTF Milestones

Original motivation: standardized way to deliver 3D into WebGL applications

Design Iteration and Multiple Implementations

Dec 2015

glTF 1.0 Spec Ratified and Released

Oct 2016

Industry Roadmap Meeting

Mar 2017

glTF 2.0 Provisional spec for industry feedback

June 2017

We Are Here!

7 Months

Microsoft joins Specification Editors

Significant Industry Adoption

Physically Based Rendering for higher-quality materials and rendering API independence
Strong glTF Momentum

- glTF is at the core of Microsoft’s 3D for Everyone vision thanks to @iamSBTron and @bgghary. Paint 3D, Viewer 3D, remix3d, Babylon, Office!

- @trigrou demoing glTF 2.0 PBR in @Sketchfab at the WebGL/WebVR/glTF meetup! @glTF3D.

- Saurabh Bhatia @saurabh Bhatia
  webGL/webVR/glTF meetup - the same glTF 2.0 asset rendered on webGL, DirectX and Vulkan!

Publicly Stated Support for glTF
glTF Ecosystem

Tools

Translators — glTF — Validator

Export

Convert | Optimize

Validator

Apps & Engines

Import

glTF in Visual Studio Code
glTF-VSCode extension

Blender DIRECT exporter in progress

Unity DIRECT Exporter in Progress

AUTODESK.
Drag and Drop FBX -> glTF
(coming soon)
http://gltf.autodesk.io/

COLLADA.
Drag and drop COLLADA -> glTF
http://cesiumjs.org/convertmodel.html

Other Translators
Assimp
OBJ2GLTF
gltf Pipeline
COLLADA2GLTF
Cesium converter

A-FRAME
Sketchfab
CESIUM
three.js
PEX
xeoEngine

All glTF spec development on open GitHub:
https://github.com/KhronosGroup/gltf

Unity DIRECT Exporter in Progress

Blender DIRECT exporter in progress

Validator

Export

Convert | Optimize

Tools

Translators — glTF — Validator

Import

Apps & Engines

http://gltf.autodesk.io/

http://cesiumjs.org/convertmodel.html


https://github.com/KhronosGroup/glTF

© Copyright Khronos Group 2017 - Page 6
What’s New in glTF 2.0

- Physically Based Rendering (PBR) material definitions
  - Material information stored in textures

- Graphics API neutral
  - Proven by engine implementations using WebGL, Vulkan and Direct3D
  - GLSL materials moved to extension for existing content and specialized use cases

- Morph Targets
  - Enhanced animation system

- Improvements
  - Binary glTF in core
  - Dozens of refinements for enhanced performance and a tighter, clearer specification

Sketchfab User: theblueturtle
https://sketchfab.com/models/b81008d513954189a063ff901f7abfe4
glTF 2.0 Scene Description Structure

- `.gltf (JSON)`
  - Node hierarchy, PBR material textures, cameras

- `.bin`
  - Geometry: vertices and indices
  - Animation: key-frames
  - Skins: inverse-bind matrices

- `.png`
- `.jpg`
  - Textures

- Geometry
- Texture based PBR materials
glTF 2.0 Scalable, Portable PBR

• glTF 2.0 PBR Requirements
  - Simple and inexpensive to implement
    - So can be everywhere - even mobile devices
  - Scalable
    - Two combinable models - but materials continue to work even if just core supported

• Metallic-Roughness Material model
  - baseColor — base color
  - metallic — metalness
  - roughness — roughness
  - Simple to implement - mandated in core

• Specular-Glossiness Material model
  - diffuse — reflected diffuse color
  - specular — specular color
  - glossiness — glossiness
  - Slightly more resources - optional extension

Illustrations by Fraunhofer
glTF 2.0 PBR - Consistency Across Engines

WebGL reference implementation
http://github.khronos.org/glTF-WebGL-PBR/

Laugh Engine running on Vulkan
https://github.com/jian-ru/laugh_engine
Industry Transitioning to glTF 2.0

• Breaking changes from 1.0 to 2.0 - but processing is streamlined and simplified
  - NOT significant work and great benefits by upgrading to 2.0

• Industry moving quickly to glTF 2.0 — lots of early adopters
  - BabylonJS, three.js, Cesium, xeogl, instant3Dhub

• `gltf-pipeline` integrating glTF 2.0 updates — has glTF 1.0 to glTF 2.0 translator
  - Open source - use this to support both glTF 1.0 and 2.0 or move your users to 2.0

• Converters/Translators/Validators glTF 2.0 updates nearly ready
  - COLLADA2GLTF and `obj2gltf` translators
  - Khronos `Validator` and `Gltf-test`

• Samples and Tutorials
  - glTF 2.0 sample models with PBR
  - Extensive glTF tutorial series in draft

Move your pipeline to glTF 2.0
Its time!
glTF Roadmap Discussions

- Incrementally ship new functionality as extensions
  - For testing out new features, or for long-term optional functionality
  - glTF baseline needs to remain easy to process and deploy

- Mesh Compression
  - Google Draco team

- Progressive Geometry Streaming
  - Fraunhofer SRC

- Basis unified compressed texture format for transmission from Binomial
  - Optimized transmission format with efficient local expansion to any GPU format

- Enhanced PBR
  - E.g. NVIDIA MDL

- Point Clouds
  - Generated by geometry capture

- Lighting Extension
  - Enhanced lighting control

- Extensions for API and language specifics
  - Optional hooks for enhanced perf/functionality
  - Vulkan, DX12, Metal, GLSL, HLSL, SPIR-V, Metal C++
Industry Calls to Action

• Implement glTF 2.0 - specification finalized!
  - https://github.com/KhronosGroup/glTF/tree/2.0/specification/2.0

• Primary glTF Online Resources
  - Github Page https://github.com/KhronosGroup/glTF
  - Resource Hub: https://www.khronos.org/gltf/

• Share and coordinate your open source glTF projects
  - https://github.com/KhronosGroup/glTF/issues/867

• glTF 2.0 Blender Exporter project complete in a few months - input and help welcome!
  - https://github.com/KhronosGroup/glTF-Blender-Exporter

• Share your roadmap priorities with us!
  - https://github.com/KhronosGroup/glTF

• Join Khronos!
  - Get directly involved in the glTF Working Group
Introduction to PBR
Intro to PBR

Diffuse

Glossy Specular

Perfect Specular

Retro-Reflective

Shadowing

Interreflection

Masking

The metallic-roughness material model is defined by the following properties:

- **baseColor** - The base color of the material
- **metallic** - The metalness of the material
- **roughness** - The roughness of the material

\[
f(l, v, h) = Diff(l, n) + \frac{F(l, h) G(l, v, h) D(h)}{4(n \cdot l)(n \cdot v)}
\]

- *l* is the light direction
- *v* is the view direction
- *h* is the half vector
- *n* is the normal
BRDF Diffuse

\[ \text{Diff}(l, n) = (1 - F(v \cdot h)) \frac{C_{\text{diff}}}{\pi} \]

Lambertian with energy conservation

\(C_{\text{diff}}\) is the diffuse reflected color. To conserve energy, the Fresnel term from specular component is subtracted from diffuse component.

\[
\begin{align*}
\text{const dielectricSpecular} &= \text{rgb}(0.04, 0.04, 0.04) \\
\text{const black} &= \text{rgb}(0, 0, 0)
\end{align*}
\]

\[C_{\text{diff}} = \text{lerp(baseColor.rgb \cdot (1 - \text{dielectricSpecular.r}), black, metallic)}\]
BRDF Specular

\[ f(l, v, h) = \text{Diff}(l, n) + \frac{F(l, h) \ G(l, v, h) \ D(h)}{4(n \ast l)(n \ast v)} \]

BRDF Specular from Cook-Torrance
BRDF Specular : F

\[ F(l, h) G(l, v, h) D(h) \]
\[ \frac{4(n \times l)(n \times v)}{4(n \times l)(n \times v)} \]

F is the Fresnel function used to simulate the way light interacts with a surface at different viewing angles.

\[ F(l, h) = F_0 + (1 - F_0) \times (1 - v \times h)^5 \]
Schlick Fernel model

\[ F_0 \] is the specular reflectance at normal incidence

\[
\text{const dielectricSpecular} = \text{rgb}(0.04, 0.04, 0.04)
\]
\[
F_0 = \text{lerp(dielectricSpecular, baseColor.rgb, metallic)}
\]
BRDF Specular : G

\[
F(l, h) \frac{G(l, v, h) D(h)}{4(n \ast l)(n \ast v)}
\]

G is the geometric occlusion derived from a normal distribution function like Smith’s function

\[
G(l, v, h) = G_1(n, l)G_1(n, v)
\]

\[
G_1(n, v) = \frac{2(n \ast v)}{(n \ast v) + \sqrt{\alpha^2 + (1 - \alpha^2)(n \ast v)^2}}
\]

\[
\alpha = (\text{roughness})^2
\]
BRDF Specular : $D$

$$F(l, h) \ G(l, v, h) \ D(h)$$

$$\frac{4(n \ast l)(n \ast v)}{4(n \ast l)(n \ast v)}$$

$D$ is the normal distribution function like GGX that defines the statistical distribution of microfacets.

$$D(h) = \frac{\alpha^2}{\pi ((n \ast h)^2(\alpha - 1) + 1)^2}$$

$$\alpha = (\text{roughness})^2$$
Metallic Roughness

PBR Materials
Specular Glossiness

PBR Materials
PBR Resources

- **Demos**
  - WebGL-PBR implementation
  - glTF 2.0 Sample Models
    - [https://github.com/KhronosGroup/glTF-Sample-Models](https://github.com/KhronosGroup/glTF-Sample-Models)

- **Articles**
  - glTF PBR Tutorial:
  - Substance PBR-guide:
    - [https://www.allegorithmic.com/pbr-guide](https://www.allegorithmic.com/pbr-guide)
  - Moving Frostbite to PBR:
  - Good example values: