Welcome to glTF Webinar | Fall 2022

Advanced PBR Material Parameters in glTF

September 20, 2022
Today’s Agenda

- Overview of the glTF PBR Material Model
  - Ed Mackey, AGI  ~20min

- Real-time implementation of transmission
  - Alex Wood, AGI  ~10min

- Expert panel Q&A session
  - Ed Mackey, AGI
  - Alex Wood, AGI
  - Nicolas Savva, Autodesk
  - Tobias Häußler, Dassault Systèmes
  - Bastian Sdorra, Dassault Systèmes
  - Emmett Lalish, Google
  - Adam Morris, Target
  - Eric Chadwick, Wayfair
How to Participate

• Speaker Questions
  - During the presentations, please submit your questions to the speakers by using the Zoom Q&A feature, not the chat button. At the end of the talk, our moderator will put as many questions as possible to the speaker.

• Recording
  - We are recording this webinar and will be sharing it via the event page on the Khronos website. A direct link will be posted in chat.

• Survey
  - To help us design future glTF events, we would appreciate it if you could complete the short survey form that will pop up at the end of the webinar. The survey link will also be sent out in our follow up email.
glTF is a “Last-Mile” Model Format

- The format is expected to be a finished product for client consumption, with model contents optimized for transmission and rendering.

- Mesh and vertex data are organized the way graphics APIs expect to receive them, for quick transfer onto the GPU.

- The core material is parameterized PBR that can work with existing PBR shaders, and is compatible with a wide array of rendering engines.
Khronos Maintains a PBR Sub-Group

- The glTF format is managed by the Khronos “3D Formats Working Group” (WG).

- There is a “Task Sub-Group” under the WG called the “PBR TSG.”

- The PBR TSG meets weekly on Mondays. Khronos membership is required.

- PBR design details discussed, often with cool 3D graphics demos.

Our Approach to Defining PBR Materials

• An extensible set of parameters based on physical properties of real materials

• Physically-based parameters must “make sense” in the real world, not target any one particular rendering strategy.

• Path tracing, ray tracing, and real-time rasterization approximations are all supported by the parameters.

• No “reference” viewer, only “sample” viewer. Real life is the reference.
Our Approach to Extending PBR Materials

- Clients may treat a PBR extension’s properties as “core,” as if they are always available on all materials, turned off by default.

- The presence of the JSON extension itself does not alter the material, because its default values intentionally do not alter material appearance.

- For example: Loading the glTF sample Box.glb in Blender yields “clearcoat” and “transmission” properties set to zero, the default.
Additional Layers via PBR Extensions

Some PBR extensions add a layer to the base material.

- Iridescence
- Sheen
- Clearcoat
Alpha Blending in glTF indicates “coverage”

- Indicates where the material is absent from a given pixel / texel.

- Not for glass or water! Zero alpha means no material, so no specular reflections from that location.

- Fractional values indicate percent coverage of the texel, for example the fringe edges of the material.

- For refractive materials such as glass and water, use transmission.
Transmission in glTF

- Allows light to pass through the material (such as water, glass) with specular and possibly refraction effects.

- Requires a different rendering strategy from alpha blending (explained later).

- Available in two modes: “Thin-walled” (the default) and volumetric (when paired with the volume extension).

- Transmitted light color is tinted by glTF’s base color, and may be blurred or sharp.
Index of Refraction

- When front & rear normals are parallel, the outgoing vector is parallel to (but offset from) the incoming vector.

- For “thin-walled” mode, there’s no thickness, so no apparent refraction. The offset is zero or infinitely small.

- However, roughness (microfacets) can perturb normals on a sub-pixel scale, causing blurriness / scattering.
Thin-walled Refraction

- Single layer of double-sided polygons guarantees front & back normal vectors are parallel with effectively no offset.
- Roughness “microfacets” can still cause sub-pixel refraction because rear facets are not parallel to front facets.
- The result is a sub-pixel scattering, blurring the transmitted light like frosted shower glass.
- The IOR and Roughness values work together to enable this “microfacet refraction.”
Volumetric Materials in glTF

- “Thin-walled” mode is turned off when “thicknessFactor” is non-zero.
- Polygon mesh expected to be watertight, enclosing a solid volume of material.
- Real-time approximations may use “thickness” (factor • texture) to estimate the travel distance through the material.
- Path tracers ignore “thickness” value and calculate this distance (but must still test for thin-walled mode).
- Transmitted color may be attenuated based on the travel distance.

AttenuationTest model CC-BY 4.0 Analytical Graphics Inc.
**Diffuse Transmission** (aka “Translucency”)

- An alternate form of transmission using the diffuse lobe instead of specular.
- The surface can be smooth and yet the transmission is completely diffuse, like a leaf or a blade of grass.
- Implies surface-level scattering of light.
- Offers its own texturable color, to indicate the scatter color along the thin surface.
- Compatible with both thin-walled and volumetric modes.

Photos by Ed Mackey from #1825
Subsurface Scattering in glTF

- Requires both volumetric (with nonzero thickness) and one of the transmission extensions, typically diffuse transmission.

- Scatters and optionally colors the light traveling inside a volume of material.
Light Transmission, Attenuation, Scattering

**Artist wants:**
- Specular reflections
- Volume-based refraction
- Thin-walled transmission
- Volume-based SSS
- Thin-walled translucency

**Specular**
- specular (F90)
- specularColor (F0)

**IOR (Refraction)**
- ior

**Volume**
- thickness
- attenuationDistance
- attenuationColor

**Transmission**
- transmission

**Subsurface Scattering**
- scatterDistance
- scatterColor

**Volume**
- thickness
- attenuationDistance
- attenuationColor

**Diffuse Transmission**
- diffuseTransmission
- diffuseTransmissionColor

**KHR_materials_transmission**
- KHR_materials_specular
- KHR_materials_ior
- KHR_materials_volume

**KHR_materials_transmission**
- KHR_materials_transmission
- KHR_materials_ior
- KHR_materials_volume

**KHR_materials_transmission**
- KHR_materials_transmission
- KHR_materials_sss
- KHR_materials_volume

**KHR_materials_diffuse_transmission**
- KHR_materials_diffuse_transmission
- KHR_materials_sss
- KHR_materials_volume
Advances in Authoring PBR Materials

Support for the new glTF PBR parameters has been added to several authoring and validation tools:

• Import + Export shipped with Blender 3.3.
• Autodesk added a glTF Material node to 3ds Max.
• UX3D added support to Gestaltor.
• The Khronos glTF Validator has upgraded support.
• VSCode’s “glTF Tools” extension upgraded support.
• MaterialX has a new <gltf_pbr> node.
MaterialX and glTF Interoperability

- Khronos contributed a `<gltf_pbr>` node to the ASWF MaterialX project. (version 1.38.4)
  Special thanks to Tobias Häußler from Dassault Systèmes for originally authoring this node.

- This node allows the glTF PBR material model to be expressed within a MaterialX node graph, for accurate import.

- One can also distill compatible MaterialX node graphs into glTF.

- glTF geometry import and material conversion work is ongoing.
  Special thanks to Bernard Kwok.
ASWF Open Chess Set “A Beautiful Game”

- Originally authored by Moeen Sayed and Mujtaba Sayed for SideFX
- Open-sourced (CC-BY 4.0) by the Academy Software Foundation (ASWF) at SIGGRAPH 2022
- High-quality USD assets with MaterialX surfaces, now distilled into glTF and its PBR
- The above screenshot is taken from a webpage running ThreeJS in real-time
Real-Time Transmission and Volume

Sample source code can be found at https://github.com/KhronosGroup/gltF-Sample-Viewer
Transmission in glTF-Sample-Viewer

```json
{
  "doubleSided": true,
  "extensions": {
    "KHR_materials_transmission": {
      "transmissionFactor": 1
    }
  },
  "name": "YellowTransRough",
  "pbrMetallicRoughness": {
    "baseColorFactor": [
      0.8468738198280334,
      0.8713662624359131,
      0.09305903315544128,
      1
    ],
    "metallicFactor": 0,
    "metallicRoughnessTexture": {
      "index": 1
    }
  }
}
```
Transmission in glTF-Sample-Viewer

- Mix diffuse BRDF with transmission BTDF based on transmission factor

Transmission = 0.0

Transmission = 1.0
Transmission in glTF-Sample-Viewer

- Render Skybox, Opaque, and Alpha Blended materials to separate Framebuffer (1024x1024 RGBA8).
- Note, images are dark because they are NOT gamma corrected.

1. Render Opaque Scene
2. Mipmap Opaque Scene
3. Render Transmissive Materials

(Gamma Corrected)
Transmission in glTF-Sample-Viewer

- Generate Mipmaps for Offscreen Opaque Scene Color Attachment.
- $f(\text{Roughness}, \text{IOR}) \rightarrow \text{Miplevel}$
  - Scale roughness with IOR where IOR=1.0 results in NO refraction and IOR=1.5 results in default amount of refraction
  - $\text{roughness} = \text{roughness} \times \text{clamp}(\text{ior} \times 2.0 - 2.0, 0.0, 1.0);$
Transmission in glTF-Sample-Viewer

• Render Skybox, Opaque, Alpha Blended, and Transmission Materials to primary Framebuffer.
• RenderState for Transmission Materials:
  - Backface cull: false
  - Depth test: true
  - Alternative approach: Render back faces to offscreen in separate pass
• Sample Offscreen Texture at UV coordinate of current fragment.

vec3 transmittedLight = textureLod(u_TransmissionFramebufferSampler, uv, lod).rgb;

1. Render Opaque Scene
2. Mipmap Opaque Scene
3. Render Transmissive Materials
Volume in glTF-Sample-Viewer

DragonAttenuation model

```
{name: "Dragon with Attenuation",
"pbrMetallicRoughness": {
  "baseColorFactor": [1, 1, 1, 1],
  "metallicFactor": 0,
  "roughnessFactor": 0
},

"extensions": {
  "KHR_materials_transmission": {
    "transmissionFactor": 1
  },
  "KHR_materials_volume": {
    "attenuationColor": [0.021, 0.64, 0.064],
    "attenuationDistance": 0.155,
    "thicknessFactor": 2.27,
    "thicknessTexture": {
      "index": 1,
      "texCoord": 0
    }
  },
  "KHR_materials_ior": {
    "ior": 1.5
  }
}
```

This work is licensed under a Creative Commons Attribution 4.0 International License
Volume in glTF-Sample-Viewer

- This should look familiar ...
- Render Skybox, Opaque, and Alpha Blended materials to separate Framebuffer (1024x1024 RGBA8).
- Generate Mipmaps

1. Render Opaque Scene
2. Mipmap Opaque Scene
3. Render Transmissive Materials
Volume in glTF-Sample-Viewer

- Render Skybox, Opaque, Alpha Blended, and Transmission Materials to primary Framebuffer.
- RenderState for Transmission Materials:
  - Backface cull: true
  - Depth test: true
- UV Sample into Offscreen Texture represents refracted view ray based on IOR (defined in KHR_materials_ior).

1. Render Opaque Scene
2. Mipmap Opaque Scene
3. Render Transmissive Materials
Volume in glTF-Sample-Viewer

- Compute the world space of the refracted vector’s exit point
  
  ```
  vec3 refractionVector = refract(-v, normalize(n), 1.0 / ior);
  vec3 worldExit = position + normalize(refractionVector) * thickness;
  vec4 ndc = projMatrix * viewMatrix * worldExit;
  vec2 uv = (ndc.xy / ndc.w + vec2(1.0)) * 0.5;
  ```

- Sample transmitted light as before:
  
  ```
  vec3 transmittedLight = textureLod(u_TransmissionFramebufferSampler, uv, lod).rgb;
  ```

- Light attenuation is computed using Beer’s Law
  
  ```
  vec3 attnCoefficient = -log(attenuationColor) / attenuationDistance;
  vec3 transmittance = exp(-attnCoefficient * transmissionDistance);
  transmittedLight = transmittance * transmittedLight;
  ```

- Mix Diffuse BRDF with Specular BTDF based on transmission factor

1. Render Opaque Scene
2. Mipmap Opaque Scene
3. Render Transmissive Materials
Transmission Summary

• glTF has brought PBR to the masses, by making the algorithms and sample implementations “approachable”.

• Many different approaches for real time approximations. Challenges in finding a balance between accuracy and performance.

• Important references to get started:
  - BRDF Implementation ([Appendix B](#) of the spec)
  - Implementation Notes in each extension.
  - References section of the extensions
  - [glTF-Sample-Models](#) repository for lots of helpful models to exercise the edge cases.
  - Sample Path Tracer maintained by Dassault:

TransmissionRoughnessTest model CC-BY 4.0 Analytical Graphics Inc.

SpecularTest model CC-BY 4.0 Analytical Graphics Inc.
Ask the Experts

Use the Zoom Q&A feature to ask your questions
Today’s Q&A Panelists

Ed Mackey (AGI)

Alex Wood (AGI)

Tobias Häußler (Dassault Systèmes)

Bastian Sdorra (Dassault Systèmes)

Nicolas Savva (Autodesk)

Emmett Lalish (Google)

Adam Morris (Target)

Eric Chadwick (Wayfair)
A recording of this webinar, along with the slides, will be available shortly at https://www.khronos.org/events/advanced-pbr-material-parameters-in-gltf