# glTF Files

This section provides additional information on the use of glTF files in Scene Composer.

## Animations

Scene Composer provides a working sample that shows how to use glTF animations. See Physically Based Rendering Solution for details.

Scene Composer supports interpolation animation; however, morphing or the channel path "weights" are not supported.

The following animation types are supported:

- translation
- rotation
- scale.

The following interpolation strategies are supported:

- linear
- step
- cubicspline

> glTF specifies rotation angles in quaternions. CGIStudio converts the values to Euler angles.

For more information about animations, see the Khronos website: animations. For a more detailed description about interpolation types see the Khronos website: animation sampler.

---

## The Software Development Kit (SDK) version

The glTF importer uses the Microsoft glTF-SDK in version v1.9.1.0. Download the Microsoft glTF-SDK from Microsoft gtTF-SDK GitHub repository.

## Reference models

Scene Composer uses the boombox as a reference model. The folder `cgi_studio_content/3D/glTF/` contains the glTF file. Visit The Khronos website: glTF Sample Models to find more available reference models.




## Supported glTF Extensions and Multi UV

Scene Composer's glTF import supports the following glTF 2.0 extensions as well as multiple UV sets (Multi UV).

| Extension | Description |
| --- | --- |
| KHR_lights_punctual | Imports point, spot, and directional lights defined in the glTF file as lights in Scene Composer. |
| KHR_materials_anisotropy | Reproduce materials with anisotropic highlights, such as brushed metal. |
| KHR_materials_clearcoat | Reproduces materials that have a clear-coat layer (clear paint). |

| | |
|---|---|
| KHR_materials_ior | Specifies the *Index of Refraction* (IOR) and improves refraction for materials such as glass. |
| KHR_materials_iridescence | Represents iridescence effects where the color changes depending on the viewing and incident angles. |
| KHR_materials_specular | Supports materials that independently control the intensity and color of specular reflection. |
| KHR_materials_unlit | Supports “unlit” materials that are not affected by lighting calculations. |
| KHR_texture_transform | Interprets UV transforms such as offset, scale, and rotation per texture. |
| KHR_materials_emissive_strength | This extension scales the emissive color or texture by a scalar *emissiveStrength* factor, allowing materials to represent much brighter emission, especially in *high dynamic range* (HDR) rendering environments. |
| KHR_materials_sheen | This extension adds a sheen layer on top of an existing *physically based rendening* (PBR) material to simulate soft, fabric-like highlights caused by fine microfibers such as cloth or velvet. |
| KHR_materials_dispersion | Represents the dispersion of light caused by differences in the index of refraction per wavelength. |
| KHR_materials_transmission | Represents the transmission of light and reproduces transparent materials such as glass. |
| KHR_materials_volume | Represents the attenuation of light inside a transmissive material (thickness, attenuation color, and attenuation distance). |

Even when UV sets other than `uv0` are specified in these extensions, Scene Composer imports multiple UV sets (Multi UV) and assigns them to the corresponding textures.

When you import a glTF file with **Import Dynamic Appearance** enabled, Scene Composer displays the properties that correspond to the extensions in the preceding table as GltfAppearance properties in the Properties panel, and you can edit them there. For the individual properties, see **Properties Overview**.

## Limitations

- Limitations in OpenGL Embedded Systems (ES) 2.0 might prevent some sample glTF files that use these extensions from producing the expected visual results in ES 2.0 environments.

    - KHR_materials_transmission: Scene Composer does not apply the transmission effect and renders the material as an opaque material.
    - KHR_materials_volume: Scene Composer applies only transmission and does not render the absorption of light.
    - KHR_materials_dispersion: Scene Composer applies only transmission and does not render dispersion.
- By design, you cannot use some combinations of extensions together. For details, check the **glTF 2.0 extension specifications** provided by Khronos.
    - For example, by design you cannot use some extensions, such as KHR_materials_anisotropy, KHR_materials_iridescence, KHR_materials_specular, KHR_materials_ior, KHR_materials_emissive_strength, and KHR_materials_sheen, together with KHR_materials_unlit.
- Meshes that use **KHR_materials_anisotropy** must have normal and tangent attributes, or the base material must have a normal map. If these conditions are not met, the anisotropy effect may not be rendered correctly.
- KHR_materials_volume and KHR_materials_dispersion require KHR_materials_transmission. When you import a glTF file that does not use KHR_materials_transmission, Scene Composer ignores these properties.

For details on the properties and parameters of each extension, refer to the glTF 2.0 extension specifications provided by Khronos.

For the behavior when you import a glTF file that uses KHR_materials_transmission, see **Importing Transmissive Materials**.

---

## Physically Based Shading (PBS)

This section describes the shader and appearance configuration that Scene Composer creates when you import a glTF file with a static Appearance. For the individual properties that apply when you import with a dynamic GltfAppearance, see **Properties Overview**.

The importer applies the metal-roughness material model to the imported 3d model. If the glTF file does not define image-based lightning (IBL) textures, the importer applies a default material. Scene Composer provides shaders for OpenGL ES 2.0 and 3.0.

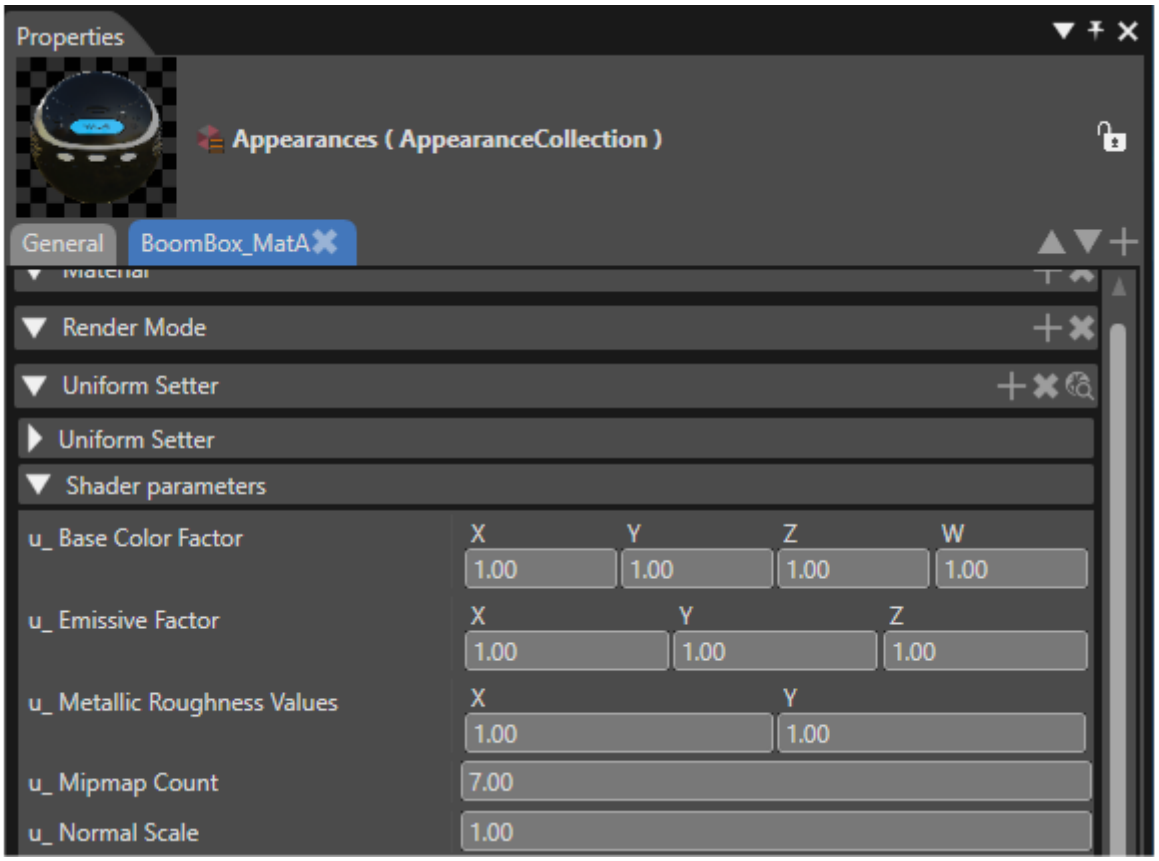
The property names used in SceneComposer differ from the glTF standard:

| Standard Property Name | Usage in SceneComposer |
|---|---|
| baseColor | u_Base Color Factor |
| metallic | determined by the X value in the u_Metallic Roughness Value |
| roughness | determined by the Y value in the |

The importer automatically sets the following appearance and shader properties, but you can also configure them manually:

## Uniforms:

| Optional | Uniform Name | Description |
|---|---|---|
| | u_MetallicRoughnessValues | holds the metallic factor in X and the roughness factor in Y. The shader multiplies these values with the metallic and roughness values from the metallic-roughness texture on a per-pixel basis. If no such texture exists, the shader uses the values from the uniform instead. |
| | u_BaseColorFactor | is a factor that the shader multiplies with the value from the albedo texture on a per-pixel basis. If no such texture exists, the shader uses the values from the uniform instead. |
| | u_MipmapCount | specifies the number of mipmaps, excluding level 0, from the specular/radiance texture. |
| optional | u_NormalScale | is a factor that the shader multiplies with the X and Y component of the normal. This uniform is available only if the fragment shader and vertex shader define `HAS_NORMALMAP`. |
| optional | u_OcclusionStrength | is a factor for the linear interpolation between the color and the color times ambient occlusion. This uniform is available only if the fragment shader and the vertex shader define `HAS_OCLUSIONMAP`. |
| optional | u_EmissiveFactor | is a factor that the shader multiplies with the value from the emissive texture on a per-pixel basis. This uniform is available only if the fragment shader and the vertex shader define `HAS_EMISSIVEMAP` |



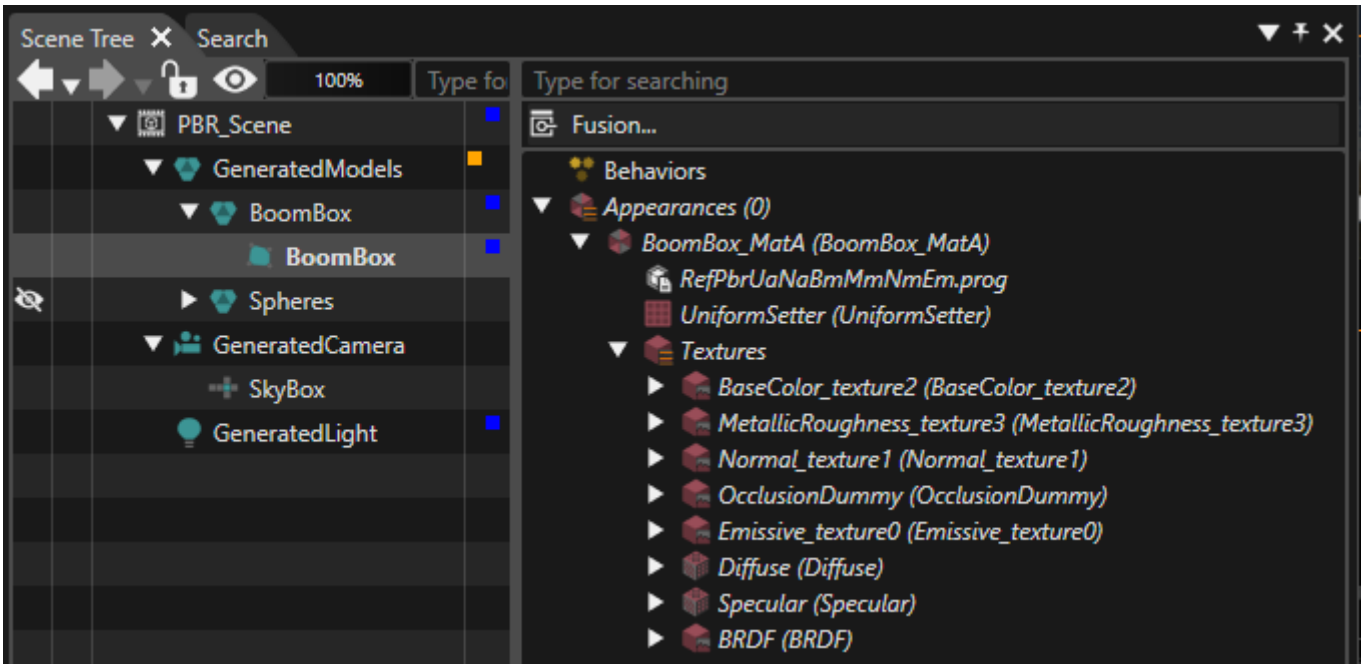

## Textures:

| Optional | Texture Name | Channels | Texture Level | Description |
|---|---|---|---|---|
| optional | albedo | RGBA | 0 | Set at texture unit 0, and define `HAS_BASECOLORMAP` in the fragment shader. |
| optional | metallic-roughness | GB | 1 | Set at texture unit 1, and define `HAS_METALROUGHNESSMAP` in the fragment shader. The metallic values must be stored in the blue channel. The roughness value must be stored in the green channel of the texture. |
| optional | tangent-space normal | RGB | 2 | Set at texture unit 2, and define `HAS_NORMALMAP` in the fragment shader and vertex shader. |

| | | | | |
|---|---|---|---|---|
| optional | occlusion | R | 3 | Set at texture unit 3, and define `HAS_OCCLUSIONMAP` in the fragment shader and vertex shader. The occlusion values are stored in the red channel of the texture. |
| optional | emissive | RGB | 4 | Set at texture unit 4, and define `HAS_EMISSIVEMAP` in the fragment shader and the vertex shader. |
| optional | Image-based lighting (IBL) | | | Provide the following textures for IBL to work correctly:<br>• diffuse radiance cubemap (see below)<br>• specular/radiance cubemap (see below)<br>• bidirectional reflectance distribution function (BRDF) (see below) |
| optional | IBL - diffuse radiance cubemap | RGB | 5 | Set at texture unit 5, and define `USE_IBL` in the fragment shader. |

| | | | | |
|---|---|---|---|---|
| optional | IBL - specular/radiance cubemap | RGB | 6 | Set at texture unit 6, and define USE_IBL in the fragment shader. Build the specular pre-convolved environment into the mipmaps. The importer automatically sets the number of mipmaps, excluding level 0, in the shader uniform *u_MipmapCount*. If you set this cubemap manually, configure this value manually. |
| optional | IBL - bidirectional reflectance distribution function (BRDF) | RG | 7 | Set at texture unit 7, and define `USE_IBL` in the fragment shader. |

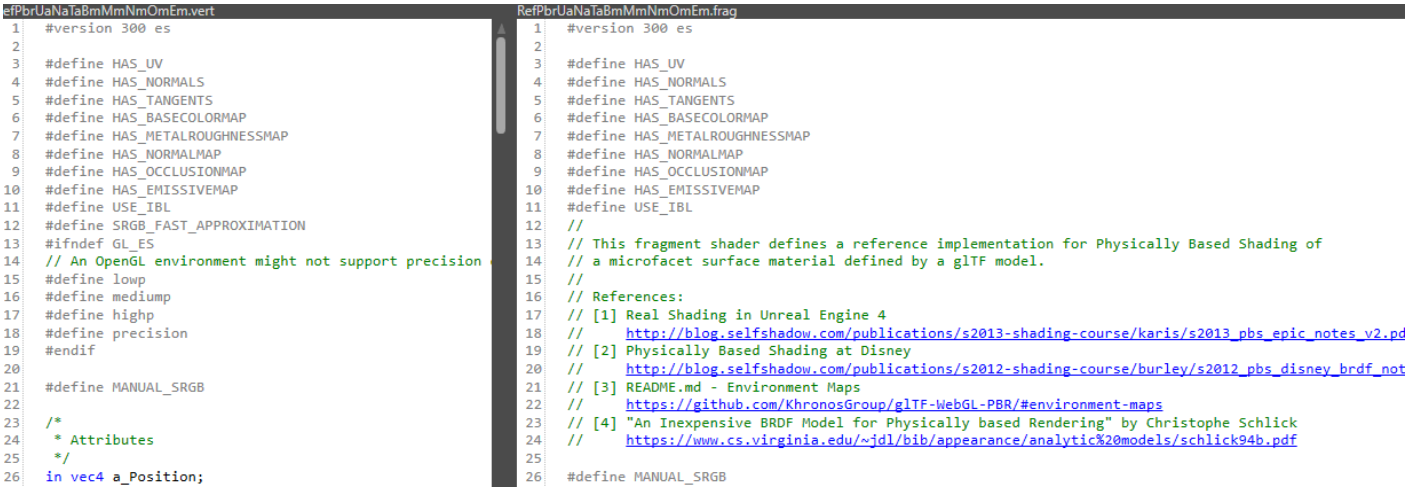
The Scene Tree shows details of the imported appearance:




## Mesh:

| Optional | Mesh Properties | Description |
|---|---|---|

| | | |
|---|---|---|
| | position | Provide the mandatory position attribute. |
| optional | normals | Define `HAS_NORMALS` in the fragment shader and vertex shader. |
| optional | tangents | Define `HAS_TANGENTS` in the fragment shader and vertex shader. |
| optional | uv set | Define `HAS_UV` in the fragment shader and vertex shader. |




*The image shows different shader variants (Click the image to enlarge it).*

## Uniform Setter:

| Optional | Uniform Setter | Description |
|---|---|---|
| | model view projection | Provide the mandatory model view projection. |
| | model matrix | Provide the mandatory model matrix |
| | normal model matrix | Provide the normal model matrix, which is mandatory if the shader defines `HAS_NORMALS` and `HAS_TANGENTS`. |

## Optimization Tips:

- Use one texture for occlusion, roughness, and metallic values, stored as an RGB texture.
- Providing tangents in the mesh avoids calculating the tangent space matrix on a per pixel basis.

For a more information about physically based rendering, see the Khronos website: metallic roughness.

# Importing Transmissive Materials

When you import a glTF file that uses KHR_materials_transmission, Scene Composer automatically creates the **multiple render pass** configuration that transmission requires.

## Render Targets That Scene Composer Creates Automatically

Scene Composer creates the following two render targets and links the offscreen render target to the onscreen render target.

| Render Target | Description |
|---|---|
| OnScreenRenderTarget | The render target that appears on the screen. The Copy camera and the Transmission camera render into this render target. |
| OffScreenRenderTarget | The render target that renders the area behind transmissive objects. The Opaque camera renders into this render target, and the transmission rendering references the result. |

## Cameras That Scene Composer Creates Automatically

Scene Composer creates three cameras in the imported scene and assigns each of them to a render target automatically.

If the glTF file contains a camera, Scene Composer creates the three cameras based on that camera and collects them into a group named `<camera name>_Group`. The camera names take no suffix (Opaque), `_Copy`, and `_Transmission`, respectively. If the glTF file contains no camera, Scene Composer creates GeneratedCamera_Opaque, GeneratedCamera_Copy, and GeneratedCamera_Transmission.

| Properties | Opaque Camera | Copy Camera | Transmission Camera |
|---|---|---|---|
| Render Target | OffScreen | OnScreen | OnScreen |
| Scope Mask | 1 | 1 | 2 |
| Clear Color Buffer | Enabled | Enabled | Disabled |
| Clear Depth Buffer | Enabled | Enabled | Disabled |

| Properties | Opaque Camera | Copy Camera | Transmission Camera |
|---|---|---|---|
| Enable Swapping | Disabled | Disabled | Enabled |
| SkyBox | Used | Used | Not Used |
| Create Mipmaps After Rendering | Enabled | Disabled | Disabled |

The Copy camera is the main camera for the on-screen display. If the glTF file contains a camera, Scene Composer sets the position and the look-at direction of the Copy camera from the transformation matrix of the glTF camera node.

> Scene Composer does not set a SkyBox on the Transmission camera. It sets a SkyBox only on the Opaque camera and the Copy camera.

The result that the Opaque camera renders into the offscreen render target is referenced when the Copy camera and the Transmission camera render. If the referenced texture has no mipmaps, aliasing occurs and the Roughness of transmissive materials does not appear as intended. For this reason, Scene Composer enables **Create Mipmaps After Rendering** automatically for the Opaque camera. For details about the property, see Properties Overview.

## Scope Mask Assignment

Scene Composer assigns a Scope Mask to each imported object automatically, depending on whether the object uses a transmissive material.

| Object | Scope Mask |
|---|---|
| Objects that do not use a transmissive material | 1 |
| Objects that use a transmissive material | 2 |

## Shaders and Uniforms

During the import, Scene Composer adds the following preprocessor definitions and uniforms to the shader, depending on the extensions that the glTF file uses.

| Extensions | Preprocessor Definitions | Uniform |
|---|---|---|
| KHR_materials_transmission | MATERIAL_TRANSMISSION<br>LINEAR_OUTPUT<br>HAS_TRANSMISSION_MAP (if a transmission texture exists) | u_TransmissionFactor<br>u_TransmissionSampler<br>u_TransmissionFramebufferSampler<br>u_TransmissionFramebufferSize |

| Extensions | Preprocessor Definitions | Uniform |
|---|---|---|
| KHR_materials_volume | MATERIAL_VOLUME<br>HAS_THICKNESS_MAP (if a thickness texture exists)<br>VOLUME_UV (if the material uses a UV set other than uv0) | u_ThicknessFactor<br>u_AttenuationDistance<br>u_AttenuationColor<br>u_ThicknessSampler |
| KHR_materials_dispersion | MATERIAL_DISPERSION | u_Dispersion |

The following describes each uniform.

| Uniform | Description |
|---|---|
| u_TransmissionFactor | The transmission factor. |
| u_TransmissionSampler | The transmission texture. |
| u_TransmissionFramebufferSampler | The texture of the offscreen render target. |
| u_TransmissionFramebufferSize | The width and height of the offscreen render target. |
| u_ThicknessFactor | The thickness of the volume. The larger the value, the stronger the absorption. |
| u_AttenuationDistance | The distance, in world units, that light travels before the volume absorbs it. The shader treats the medium as homogeneous over this attenuation distance. |
| u_AttenuationColor | The color of the light that the volume absorbs. This color determines the tint of the volume material. |
| u_ThicknessSampler | The thickness texture of the volume. |
| u_Dispersion | The strength of the dispersion. A value of 0.0 means no dispersion. The larger the value, the stronger the iridescent refraction. |

> For the **Transmission**, **Volume**, and **Dispersion** properties that apply when you use GltfAppearance, see **Properties Overview**.

---

# Useful links

Khronos references include the following resources:

- glTF Overview (https://www.khronos.org/gltf/)
- glTF 2.0 Specification (https://github.com/KhronosGroup/glTF/tree/master/specification/2.0)
- glTF 2.0 Sample Models (https://github.com/KhronosGroup/glTF-Sample-Models/tree/master/2.0)

The Khronos specification is available here:

- Specification (https://github.com/KhronosGroup/glTF)

The following viewers are available:

- BabylonJS Sandbox (https://sandbox.babylonjs.com/)
- donmccurdy glTF viewer (https://gltf-viewer.donmccurdy.com/)
- Khronos glTF Sample Viewer (https://github.khronos.org/glTF-Sample-Viewer-Release/)

---

Revision #24
Created 2023-02-14 02:14:39 UTC by Tsuyoshi.Kato
Updated 2026-09-11 12:50:17 UTC by Elisabeth Zauner