Real-time Asset Creation Guidelines

Version 1.0.0
Last Updated: July 28, 2020

Content

Real-time Asset Creation Guidelines Summary

Asset Creation Guidelines Summary

Executive Summary

The goal of these guidelines is to enable artists to streamline the creation of 3D assets that can be easily and reliably used by merchants for real-time rendering on multiple delivery platforms.

This summary is a preview of an upcoming full set of guidelines that will contain many more details, together with example 3D asset creation workflows using popular 3D tools. This summary has been released to enable feedback and suggestions from the industry to guide and help improve the content of the full guidelines.

These guidelines are primarily intended for 3D artists who are familiar with polygonal 3D workflows but who may not be familiar with creating assets for 3D web/mobile delivery. They contain best practices and modeling standards for high-quality, efficient 3D assets that will be performant in augmented reality (AR) and virtual reality (VR) experiences, product configurators, and interactive web-based 3D marketing tools.

We've designed the guidelines to be DCC (Digital Content Creation) tool agnostic as the principles should be applicable to any 3D asset creation software. Selected industry-vertical and software specific workflows, including creating 3D assets from true-geometry CAD models will be covered by upcoming Asset Creation Workflows in the full version of the guidelines.

File Formats and Asset Structure Summary

When building 3D assets, it is important to export the final product models into widely recognized file formats and to structure data within those files using common conventions.

glTF is a royalty-free, open standard file format for 3D assets that is widely adopted by 3D authoring tools and viewers on diverse platforms. glTF enables asset materials to use Physically Based Rendering (PBR) for realistic visual product representations. glTF assets are represented as .gltf files with referenced textures & geometry, or binary .glb files that embed the textures directly instead of referencing them as external images.

iOS devices do not natively support glTF but use Apple's proprietary USDz format. glTF and USDz have similar capabilities but some glTF features are not supported by USDz. We will cover the differences in the full version of the Real-time Asset Creation Guidelines.

To support these differences we recommend using the Publishing Targets workflow. Using this system a Source Asset is created with the highest quality content in PBR format. This asset can then be decimated/simplified into different targets, to support a variety of viewers, each supporting different material features. For example, glTF supports the use of an Emissive value or texture, whereas USDz only supports an Emissive texture.

- Publishing Targets should be created with the smallest file size that does not unacceptably degrade visual quality, to reduce download time and provide a smoother guest experience. See Publishing Targets Summary section.
- Most assets should be placed so the center bottom is at world coordinate 0,0,0.
- Articulated assets should have specifc pivot placement to direct movement and animation.
- Common characters should be used when naming asset files: a-z, _, -, 0–9.
- Start the file name with a letter as the first character.
- Use underscore _ or hyphen - to separate words within a file name, rather than blanks.

Coordinate Systems and Scale Summary

DCC 3D asset authoring tools use a variety of internal coordinate systems and measurement units. It is important to understand the differences between modeling coordinates and normal vector coordinates.

- Modeling coordinates: both glTF and USDz use the right-handed coordinate system, with `+Y` as world up, and the front of the asset facing `+Z`. Positive rotation is counterclockwise.
- Normal vector coordinates: glTF and USDz use OpenGL conventions where `+X` is right, `+Y` is up, and `+Z` points toward the viewer. glTF supports mikktspace tangent space.
- Scale: use 1 unit as 1 meter when possible. If you use non meter-based units, applying appropriate multipliers will be necessary. E.g. 1 inch = 0.0254 meters.

Geometry Summary

It is recommended to design 3D assets with high accuracy geometry, which can be used for high quality product renderings, and later distill that geometry into smaller assets optimized for real-time deployment. To deliver compelling experiences on the web and on mobile devices, delivery-

ready assets should be optimized for quick loading using the following techniques:

- The asset should be visually identical to the reference photos and built to real-world scale.
- The asset should use the minimum amount of geometry to achieve visual realism. Small surface geometric details should be baked into normal maps whenever possible. See Publishing Target Summary section for triangle count guidance for different platforms.
- Avoid nGons when possible. Quads (4-sided mesh) are recommended for source models, using triangulation only for final asset optimization. Be sure to keep non-triangulated meshes on hand for authoring future revisions to the model.
- Avoid single vertex points that have a large number of edges connected to them (i.e. high valence vertices with 10+ edges connected to a single point).
- Make sure there are no shading errors on the asset which are indicative of an asset problem, such as two faces overlapping or vertex normals needing to be recalculated.
- Smooth edges or bevels can refine edge transitions for increased realism in an aesthetically pleasing way. A common practice is to add extra edge loops or to use normal maps to achieve similar effects.
- There should not be any obvious holes, unintentional visible gaps, or non-manifold geometry in the asset.
- For better rendering results with assets that have transparent components, use separate meshes and materials for transparent vs. opaque parts.
- Reset any transform data, construction history, and modifier stacks to avoid export discrepancies.

UV Coordinates Summary

An asset optimized for web and mobile experiences should have fully unwrapped UVs for all parts. High quality UVs are required to create convincing real-time assets that faithfully represent real-world products.

- UV layouts can be tiled or atlased, depending on the asset. When surface details must be unique an atlas layout is best. However if surfaces use repetitive patterns then tiling textures are better.
- UV seams should be placed on natural breaks of the real-world asset, or hidden in less-visible areas.
- Use contiguous UV elements as much as possible, minimize small UV shells, and endevour to reduce visible texture stretching or warping.
- Tiled UVs should use Real-World Scale (RWS) and be normalized to ensure an even texture density. However, detailed asset components such as zippers or logos may use larger UVs to maintain sufficient resolution and avoid blurriness.
- Tiled UVs require an additional set of non-tiled UVs in atlas layout, for an ambient occlusion texture.
- UV atlas layouts should have all UVs positioned within the 0-1 space, avoid overlapping, and minimize wasted empty areas.

Materials Summary

Typically textures and materials make up a larger percentage of an asset file size than geometry. Materials optimized for visually realistic real-time rendering should use a physically-based rendering (PBR) workflow. PBR materials help minimize file sizes, enable a wide range of surface types, are easy to use and understand, and use less memory when rendering. Recommended guidelines for real-time-optimized materials include:

- Use PBR materials in Metallic-Roughness format.
- Use the minimum the number of materials necessary for sufficient visual asset realism.
- Semi-transparent parts of the asset should use a separate material from any opaque parts, and the meshes should be separated as well.
- Each characteristic of a material should use no more than one texture, and should not use blending nor layering.
- Use values instead of textures whenever a texture would be a solid single value, as this can significantly reduce file sizes.

Textures Summary

Use high quality texture assets when creating the high-resolution Source Asset. We recommend saving the Source textures in PNG format rather than compressed JPG format. Textures can then be down-sampled into PNG or JPG formats and different resolutions to support different publishing targets, which may require differing file sizes and material features.

PBR texture types include:

- Base Color controls the colors you see when a surface is evenly lit, without reflection in the way. In a PBR Metalness-Roughness material, the Base Color stores reflection color for metals (gold, copper, brass, etc.). When the surface is non-metal, Base Color is used for traditional non-reflective diffuse texture (wood, brick, fabric, etc.).
- Alpha Coverage is used for transparent details, and is always stored in the alpha channel of the Base Color texture. Alpha Coverage represents the overall visibility of the surface; unlike glass or liquids it dims all surface details including reflections. Typical uses include surfaces with many small gaps such as wicker or burlap. Glass can be represented (imperfectly) by forcing 30% opacity; this will be replaced in the near future with proper transmission and refraction via new glTF extensions.
- ORM is a combination texture for storing Ambient **O**cclusion in the red channel, **R**oughness in the green channel, and **M**etalness in the blue channel.
- Ambient occlusion is used for soft shadows wherever model intersections and crevices occur. For best results, use a raytraced renderer to precalculate occlusion and store this in an Occlusion texture. This is best accomplished when the model and material are near completion, so all the details can affect the occlusion calculations.
- Roughness defines the micro-surface (small) bumpiness, which essentially controls how blurry

or sharp reflections will be.

- Metalness controls which parts of a surface are considered metallic or not. Metallic surfaces reflect light very differently from non-metals. Metalness has a large effect on the final color and reflectivity of the surface, as well as informing what color and texture should be stored in Base Color and Roughness.
- Normal bump is used for macro-surface (large) bumpiness. Normal adds variation in surface direction to simulate grooves, pits, fibers, etc. Normal can be used to store the curvature from a higher-detail model, allowing a lower-resolution model to look like it has more smoothness or detail.
- Emissive can be used for internal lighting, glow-in-the-dark paint, LED displays, etc. For best results, use a raytraced renderer to precalculate emissive light bounces and store this in an Emissive texture. This is best accomplished when the model and material are near completion, so all the details can affect the emissive calculations.

Recommended guidelines for real-time textures include:

- Use sRGB color space for Base Color and Emissive textures, use Linear color space for all other texture types.
- Tangent-space normal maps for glTF and USDz should use the OpenGL convention (red right, green up) and the mikktspace tangent space.
- When a material requires transparency, use PNG for the Base Color and store the Alpha Coverage in the alpha channel.
- Down-sampled textures for publishing targets should be as small as possible without degrading visual details. Use JPG textures as much as possible for Base Color (without transparency) and Emissive. If JPG artifacts are too extreme, use PNG textures. See Publishing Target Summary section below for format and resolution recommendations.

Lighting Summary

Common types of lighting techniques include:

- Image-based Lighting (IBL) uses a panoramic environment image, used for both specular reflections (glossy surfaces) and soft diffuse lighting (rough surfaces). IBL textures can be created from panoramic high-dynamic range photography or rendered from a computer graphics scene. The Khronos sample GLTF viewer can use IBL.
- Emissive is used to make surfaces glow, as if they are lit internally. Emissive maps usually do not cast light onto other surfaces and can be a texture or just a solid color value.

Recommended guidelines for real-time-optimized lighting include:

- The final exported 3D asset should not include any lighting, including no dynamic lighting (i.e. Direct, Spot, or Point lights)
- Do not use baked lighting for product assets as the technique doesn’t support dynamic lighting or a PBR workflow.

- Do not use a single ambient color light to light a scene during authoring, as Ambient lights are not physically based.
- Use a consistent Image Based Lighting setup to test and confirm the accuracy of your models, materials and textures.

Publishing Targets Summary

We recommend creating a Source Asset with the highest quality content in PBR format, using high-resolution geometry and textures. This asset can be used as a ``single source of truth'' which is then decimated/simplified into different targets, to support a variety of viewers, each supporting different material features.

To optimize design decisions when creating real-time 3D assets, it is vital to understand the capabilities and limitations and of the target delivery platforms. Visual fidelity must be balanced with real-time performance - which must be tested on-device. The following are general `lowest common denominator' guidelines for desktop Web and mobile AR platforms.

For each publishing target:

- File Size: Ideally less than 5MB. Note that as glTF geometry and texture compression extensions, such as glTF Universal Textures using the KTX container and geometry compression using Draco, on the GLTF roadmap become widely available, smaller assets or more visual fidelity at the same asset size will be possible.
- Draw calls: should be minimized by consolidating meshes, and using fewer materials.
- Triangle Count: 100,000 triangles or less.
- Texture Aspect Ratio: use power of 2 resolutions, square aspect ratio is not required.
- Texture Size: Use `1024*1024` (1K) or `2048*2048` (2K) for BaseColor, ORM and Emissive maps. 2K is recommended for Normal maps which are more sensitive to reduced resolutions than even Albedo maps. Normals Maps are also severely sensitive to JPG artifacts - a 2K JPG giving the same quality as same as 1K PNG normal map.