Skip to content

Universal Scene Description

Importing USD Files

USD files typically represent the scene as a hierarchy of primitives, or prims. Individual prims contain data to describe scene entities, such as geometry, lights, cameras and transform hierarchies. Blender's USD importer converts USD prims to a hierarchy of Blender objects. Like the USD exporter, the importer does not yet handle certain USD composition concepts, such as layers and references.

The following USD data types can be imported as Blender objects:

- Cameras
- Curves
- Lights
- Materials
- Meshes
- Point Clouds
- Primitive Shapes
- Volumes

For more information on how the various data types are handled, see the following descriptions of the Import Options.

Note

When importing a USDZ archive, it is important to carefully consider the Import Textures option to determine whether and how to copy texture files from the zip archive.

Xform and Scope Primitives

USD provides an `Xform` prim type, containing transform data, which can be used to represent transform hierarchies and to organize the scene. Such `Xform` prims are imported as Blender empty objects.

USD also supports `Scope` primitives, which are entities that do not contain transform data, but which serve to group other element of the scene. Blenc doesn't have an exact counterpart to the concept of a scope, so such primitives are imported as Blender empties located at the origin. This is an imperfe representation, because empty objects have a transform and `Scopes` do not, but this approach nonetheless helps preserve the structure of the scene hierarchy.

PointInstancer Primitives

USD provides a `UsdGeomPointInstancer` prim type, containing instances that are scattered on a primitive's points.

These are imported into Blender as Point Clouds using a Geometry Nodes Modifier and the Instance on Points Node.

Animations

The importer supports two types of animation:

- **Animating transforms**: If a USD primitive has time-varying transform data, a Transform Cache constraint will be added to the imported Blender object.
- **Animating geometry**: Animated mesh, curve, and point cloud geometry is supported by adding a Mesh Sequence Cache modifier to the imported data. Geometry attribute (USD Primvar) animation is supported for all data types which have corresponding Blender equivalents. This includes colors UVs, velocities, and other generic attribute data. Note that USD file sequences (i.e. a unique file per frame) are not supported.

Materials

If a USD mesh or geometry subset has a bound material, the importer will assign to the Blender object a material with the same name as the USD materi If a Blender material with the same name already exists in the scene, the existing material may be used, depending on the Material Name Collision option

Otherwise, a new material will be created.

If the USD material has a USD Preview Surface shader source, the Viewport Display color, metallic, and roughness are set to the corresponding USD Preview Surface input values.

There is also an *Import USD Preview* option to convert USD Preview Surface shaders to Blender Principled BSDF shader nodes. This option can be lossy, as it does not yet handle converting all shader settings and types, but it can generate approximate visualizations of the materials.

Coordinate System Orientation

If the imported USD is Y up, a rotation will be automatically applied to root objects to convert to Blender's Z up orientation.

Import Options

The following options are available when importing from USD:

General

Path Mask
: Import only the subset of the USD scene rooted at the given primitive.

Include

Visible Primitives Only
: Do not import invisible USD primitives. Only applies to primitives with a non-animated visibility attribute. Primitives with animated visibility will always be imported.

Defined Primitives Only
: When disabled this allows importing USD primitives which are not defined, such as those with an override specifier.

Set Frame Range
: Update the scene's start and end frame to match those of the USD stage.

Create Collection
: Add all imported objects to a new collection.

Relative Path
: Select the file relative to the blend-file.

Apply Unit Conversion Scale
: Scale the scene objects by the USD Stage `metersPerUnit` value. This scaling is applied in addition to the value specified in the Scale optio

Scale
: Value by which to scale the imported objects in relation to the world's origin.

Light Intensity Scale
: Scale for the intensity of imported lights.

Custom Properties
: Behavior when importing USD attributes as Custom Properties.

None:
: Does not import USD custom attributes.

User:
: Imports USD attributes in the `userProperties` namespace as custom properties. The namespace will be stripped from the property names.

All Custom:
: Imports all USD custom attributes as custom properties. Namespaces will be retained in the property names.

Object Types

Cameras

Import `UsdGeomCamera` primitives as Camera Objects (perspective and orthographic).

Curves

Import `UsdGeomBasisCurves` primitives as Curves and `UsdGeomNurbsCurves` as Blender meshes.

Lights

Import lights as Light Objects. Does not currently include cylinder or geometry lights.

World Dome Light

Converts the first discovered `UsdLuxDomeLight` dome light to a world background shader.

Materials

Import UsdPreviewSurface materials.

Meshes

Import `UsdGeomMesh` primitives as Mesh Objects.

Volumes

Import `UsdVolVolume` OpenVDB assets as Volume Objects.

Point Clouds

Import `UsdGeomPoints` primitives as Point Cloud Objects.

USD Shapes

Import USD primitive shapes as Blender meshes. `UsdGeomCapsule`, `UsdGeomCylinder`, `UsdGeomCone`, `UsdGeomCube`, a `UsdGeomSphere` are supported.

Display Purpose

Render

Include primitives with purpose `render`.

Proxy

Include primitives with purpose `proxy`.

Guide

Include primitives with purpose `guide`.

Material Purpose

Attempt to import materials with the given purpose. If no material with this purpose is bound to the primitive, then the fallback behavior, if any, is noted below.

All Purpose:

Attempt to import `allPurpose` materials.

Preview:

Attempt to import `preview` materials. Load `allPurpose` materials as a fallback.

Full:

Attempt to import `full` materials. Load `allPurpose` or `preview` materials, in that order, as a “fallback.”

Geometry

UV Coordinates

Read mesh UV coordinates.

Color Attributes

Convert the USD mesh `displayColor` values to Blender’s Color Attributes.

Mesh Attributes

Read USD `Primvars` as mesh attributes.

Subdivision

Create Subdivision Surface modifiers based on the USD `SubdivisionScheme` attribute.

Validate Meshes

Check the imported mesh for corrupt data and fix it if necessary. When disabled, erroneous data may cause crashes displaying or editing the meshes. This option will make the importing slower but is recommended, as data errors are not always obvious.

Merge parent Xform
: Allow USD primitives to merge with their Xform parent if they are the only child in the hierarchy.

Rigging

Shape Keys
: Imports USD blend shapes as Blender's Shape Keys.

Armatures
: Imports USD skeletons as Blender's Armatures.

Materials

Import All Materials
: Also import materials that are not used by any geometry. Note, when this option is false, materials referenced by geometry will still be imported.

Import USD Preview
: Convert USD Preview Surface shaders to Principled BSDF shader networks.

Set Material Blend
: If the *Import USD Preview* option is enabled, the material blend method will automatically be set based on the `opacity` and `opacityThreshold` shader inputs, allowing for visualization of transparent objects.

Material Name Collision
: Behavior when the name of an imported material conflicts with an existing material.

Make Unique:
: Import each USD material as a unique Blender material.

Reference Existing:
: If a material with the same name already exists, reference that instead of importing.

Textures

When importing a USDZ package, the following options specify whether and how texture asset dependencies of the USD should be copied from the zip archive so they can be loaded into Blender.

Import Textures
: Behavior when importing textures from a USDZ archive.

None:
: Don't import textures. Note that, with this option, material textures may fail to be resolved in Blender.

Packed:
: Import textures as packed data in the Blender file.

Copy:
: Copy files to the directory specified in the **Textures Directory** option.

Textures Directory
: Path to the directory where imported textures will be copied, when the **Import Textures** mode is **Copy**.

Note that the default textures directory is the relative path `//textures`, which requires the Blender file to have been saved before importing, the relative path can be resolved.

File Name Collision
: Behavior when the name of an imported texture file conflicts with an existing file.

Use Existing:
: If a file with the same name already exists, use that instead of copying.

Overwrite:

Overwrite existing files.

Particles and Instancing

Scene Instancing

Import USD scene graph instances as collection instances, otherwise they are imported as copies.

Exporting to USD Files

Universal Scene Description (USD) files can contain complex layering, overriding, and references to other files. Blender's USD Exporter takes a much simpler approach. When exporting, all visible, supported objects in the scene are exported, optionally limited by their selection state. Blender does not (yet) support exporting invisible objects, USD layers, variants, etc.

The following objects can be exported to USD:

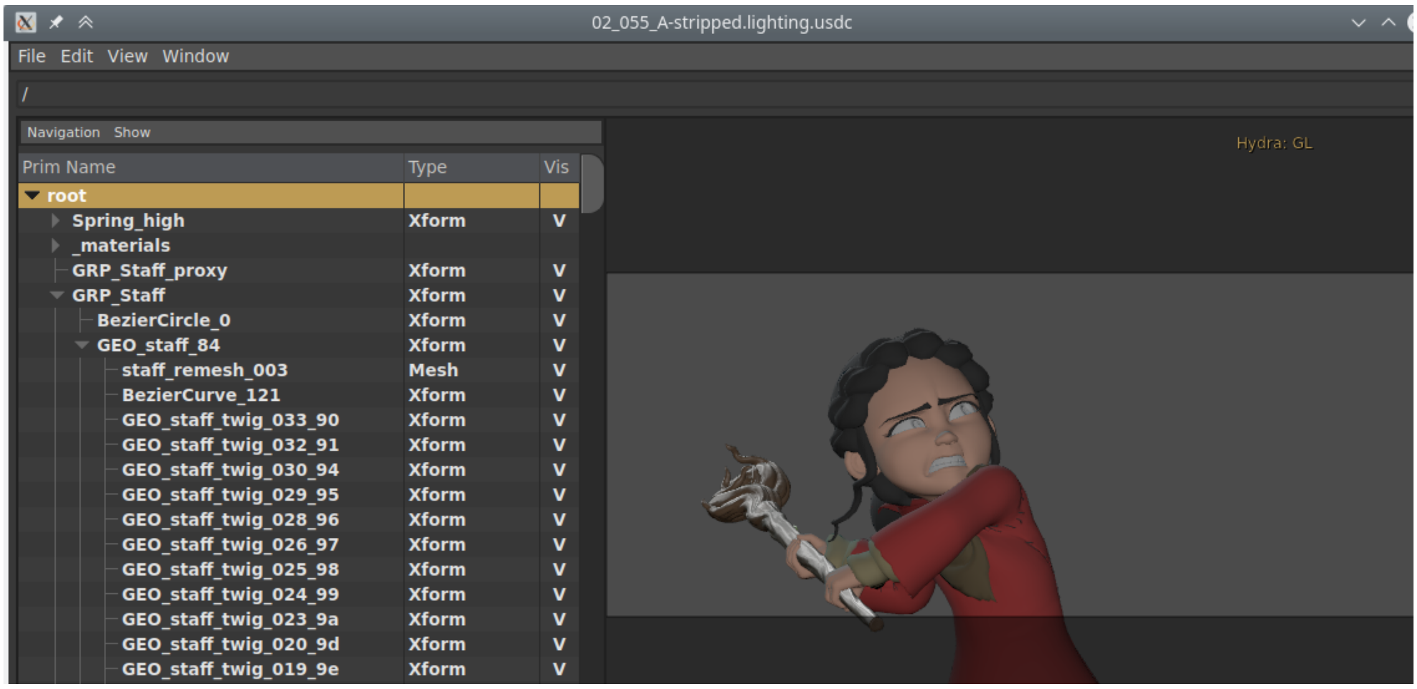
- Meshes (of different kinds, see below).
- Cameras (perspective cameras only at the moment, not orthogonal ones).
- Curves
- Lights
- Hair (exported as curves, and limited to parent strands).
- Point Clouds
- Volumes
- Armatures

When exporting an animation, the final, evaluated mesh is written to USD. This means that the following meshes can be exported:

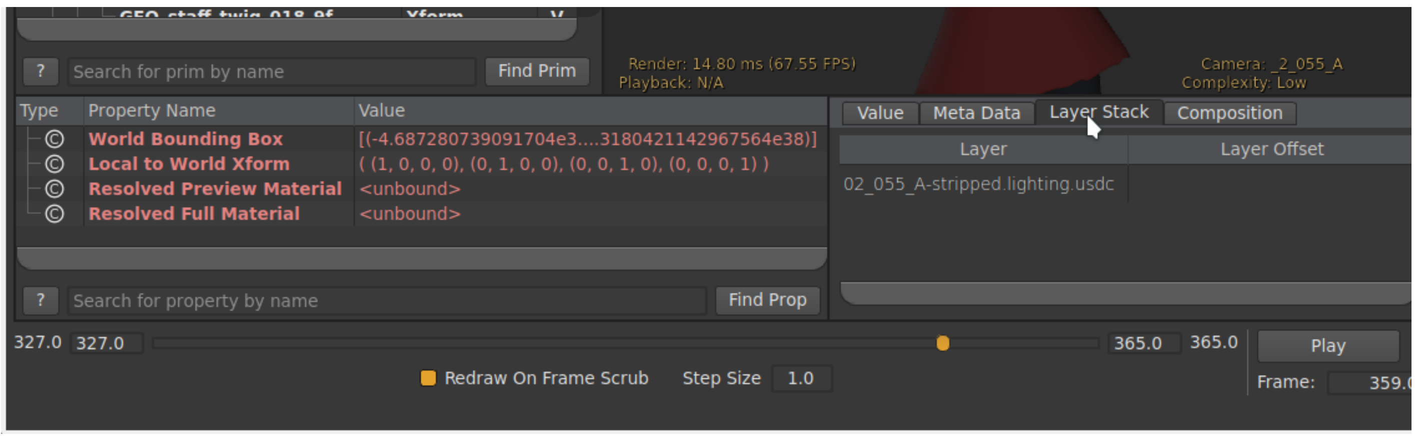
- Static meshes.
- Deforming meshes; here the topology of the mesh does not change, but the locations of the vertices change over time. Examples are animated characters or bouncing (but not cracking) objects.
- Arbitrarily animated meshes; here the topology does change. An example is the result of a fluid simulation, where splashes of fluid can break off the main body.
- Metaballs are exported as animated meshes.

Note

To export the Blender scene as a USDZ archive, set the file extension of the output file to `.usdz`. The exported USDZ package will be a zip archive containing the USD and its texture file dependencies.



Prim Name	Type	Vis
root		
Spring_high	Xform	V
_materials		
GRP_Staff_proxy	Xform	V
GRP_Staff	Xform	V
BezierCircle_0	Xform	V
GEO_staff_84	Xform	V
staff_remesh_003	Mesh	V
BezierCurve_121	Xform	V
GEO_staff_twig_033_90	Xform	V
GEO_staff_twig_032_91	Xform	V
GEO_staff_twig_030_94	Xform	V
GEO_staff_twig_029_95	Xform	V
GEO_staff_twig_028_96	Xform	V
GEO_staff_twig_026_97	Xform	V
GEO_staff_twig_025_98	Xform	V
GEO_staff_twig_024_99	Xform	V
GEO_staff_twig_023_9a	Xform	V
GEO_staff_twig_020_9d	Xform	V
GEO_staff_twig_019_9e	Xform	V


Render: 14.80 ms (67.55 FPS)
Playback: N/A

Camera: _2_055_A
Complexity: Low

Type	Property Name	Value
©	World Bounding Box	[(-4.687280739091704e3....3180421142967564e38)]
©	Local to World Xform	((1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1))
©	Resolved Preview Material	<unbound>
©	Resolved Full Material	<unbound>

Layer	Layer Offset
02_055_A-stripped.lighting.usdc	

Shot from Spring exported to USD and opened in USDView.

Export Options

The following options are available when exporting to USD:

General

Root Prim

If set, add a transform primitive with the given path to the stage as the parent of all exported data.

Include

Selection Only

When checked, only selected objects are exported. Instanced objects, for example collections that are instanced in the scene, are consider 'selected' when their instancer is selected.

Visible Only

Only exports objects that are not hidden. Invisible parents of exported objects are exported as empty transforms.

Animation

When checked, the entire scene frame range is exported. When unchecked, only the current scene frame is exported.

Blender Data

Custom Properties

Exports Custom Properties as USD attributes. The *Namespace* property is used to determine the namespace that the attributes are written to.

Namespace

If set, add the given namespace as a prefix to exported custom property names. This only applies to property names that do not already ha a prefix (e.g., it would apply to name `bar` but not `foo:bar`) and does not apply to Blender object and data names which are always exported in the `userProperties:blender` namespace.

By default, `userProperties` namespace is used.

Blender Names

Author USD custom attributes containing the original Blender object and object data names.

Allow Unicode

Preserves UTF-8 encoded characters when writing USD prim and property names (requires software utilizing USD 24.03 or greater when openi the resulting files).

File References

Relative Paths

Use relative paths to reference external files (i.e. textures, volumes) in the exported USD file, otherwise use absolute paths.

Convert Orientation

Convert orientation axis to a different convention to match other applications. Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Forward / Up Axis

By mapping these to different axes you can convert rotations between applications default up and forward axes.

Units

Set the USD Stage `metersPerUnit` metadata to the chosen measurement.

Meters Per Unit

Value to use for `metersPerUnit` if *Custom* Units are selected.

Xform Ops

The type of transform operators to use to transform prims.

Translate, Rotate, Scale:
Export with translate, rotate, and scale `Xform` operators.

Translate, Orient, Scale:
Export with translate, orient quaternion, and scale `Xform` operators.

Matrix:
Export matrix operator.

Use Settings for

Determines whether to use *Viewport* or *Render* visibility of objects, modifier settings, and other properties providing similar options, during expor

Object Types

Meshes

Exports Mesh Objects

Lights

Exports Light Objects The `UsdLuxShapingAPI` is used to support spot lights.

World Dome Light

Convert the world material to a `UsdLuxDomeLight`. Currently works for simple materials, consisting of an environment texture connected t a background shader, with an optional vector multiply of the texture color.

Cameras

Exports Camera Objects Only perspective cameras are exported.

Curves

Exports Curve Objects

Point Clouds

Exports Point Cloud Objects

Volumes

Exports Volume Objects

Hair

Exports parent hair strands are exported as a curve system. Hair strand colors are not exported.

Note

The corresponding USD schema type used during Export is analagous to the type read during Import. See the Import section for details.

Geometry

UV Maps

When checked, includes UV coordinates for exported meshes. The name of the UV map in USD is the same as the name in Blender.

Rename UV Maps

Exports UV maps using the USD default name (`st`) as opposed to Blender's default name (`UVMap`).

Normals

When checked, includes normals for exported meshes. This includes custom loop normals.

Merge parent Xform

Merge USD primitives with their Xform parent if possible. USD does not allow nested `UsdGeomGprims` , intermediary Xform prims will be defined to keep the USD file valid when encountering object hierarchies.

Triangulate

Triangulates the mesh before writing. For more detail on the specific option see the Triangulate modifier.

Rigging

Shape Keys

Export shape keys as USD blend shapes.

Absolute shape keys are not supported.

Armatures

Export Armatures and meshes with Armature Modifiers as USD skeletons and skinned meshes.

Limitations:

- Modifiers in addition to Armature modifiers will not be applied.
- Bendy bones are not supported.

Only Deform Bones

Only export deform bones and their parents.

Materials

Exports material information of the object. By default the exporter approximates the Principled BSDF node tree by converting it to USD's Preview Surface format.

When a mesh has multiple materials assigned, a geometry subset is created for each material. The first material (if any) is always applied to the mesh itsel as well (regardless of the existence of geometry subsets), because the Hydra viewport does not support materials on subsets. See USD issue #542 for more information.

Note

If *USD Preview Surface Network* and *MaterialX Network* are disabled, the material is set to the viewport materials of meshes.

Displacement Support

Displacement is support with some caveats:

- Only object space displacement is supported (no vector displacement).
- *Midlevel* and *Scale* controls can only be constants.
- MaterialX is currently not supported, see the feature commit for details.

USD Preview Surface Network

Approximates a Principled BSDF node tree by converting it to USD's Preview Surface format.

Note

To support `opacityThreshold` , sometimes known as "Alpha Clip", the node tree must either use a Math node set to `Round` , if the desired threshold is 0.5, or by using a pair of Math nodes implementing `1 - (value < threshold)` . The result should be plugged int the Alpha socket on the Principled BSDF node.

Warning

Not all nodes are supported; currently only simple node trees containing Diffuse BSDF, Principled BSDF, Image Textures, UVMap, and Separate RGB nodes are supported.

MaterialX Network

Generates material shading graphs using the MaterialX standard. This standard is designed to support a high amount of interoperability among DCCs <Digital Content Creation>. In Blender, MaterialX supports most of the shader nodes and their functionality but has a few caveats (see below).

Implementation Caveats

When using the Principled BSDF, the resulting graph is very usable. However, when using some of the other BSDFs, some of the generated shading graphs are difficult for other DCC's to understand.

Export Textures

Method for exporting textures.

Keep:
Use original location of textures.

Preserve:
Preserve file paths of textures from already imported USD files. Export remaining textures to a 'textures' folder next to the USD file.

New Path:
Export textures to a 'textures' folder next to the USD file.

Overwrite Textures

Allow overwriting existing texture files when exporting textures.

USDZ Texture Downsampling

Choose a maximum size for all exported textures.

Keep:
Keep all current texture sizes.

256:
Resize to a maximum of 256 pixels.

512:
Resize to a maximum of 512 pixels.

1024:
Resize to a maximum of 1024 pixels.

2048:
Resize to a maximum of 2048 pixels.

4096:
Resize to a maximum of 4096 pixels.

Custom:
Specify a custom size.

USDZ Custom Downscale Size
The size in pixels of the *Custom* downsampling.

Experimental

Instancing

As this is an experimental option. When unchecked, duplicated objects are exported as real objects, so a particle system with 100 particles that is displayed with 100 meshes will have 100 individual meshes in the exported file. When checked, duplicated objects are exported as a reference to the original object. If the original object is not part of the export, the first duplicate is exported as real object and used as reference.

Exporter Limitations

Single-sided and Double-sided Meshes

USD seems to support neither per-material nor per-face-group double-sidedness, so Blender uses the flag from the first material to mark the entir mesh as single/double-sided. If there is no material it defaults to double-sided.

Materials

When there are multiple materials, the mesh faces are stored as geometry subset and each material is assigned to the appropriate subset. If there is only one material this is skipped. Note that the geometry subsets are not time-sampled, so it may break when an animated mesh changes topology

Hair

Only the parent strands are exported, and only with a constant color. No UV coordinates, and no information about the normals.

Camera

Only perspective cameras are exported.

Particles

Particles are only written when they are alive, which means that they are always visible. There is currently no code that deals with marking them as invisible outside their lifespan.

Objects instanced by particle system are exported by suffixing the object name with the particle's persistent ID, giving each particle transform a unique name.

Instancing/Referencing

This is still an experimental feature that can be enabled when exporting to USD. When enabled, instanced object meshes are written to USD as references to the original mesh. The first copy of the mesh is written for real, and the following copies are referencing the first. Which mesh is considered 'the first' is chosen more or less arbitrarily.

USDZ

Due to a current limitation in the USD library, UDIM textures cannot be include in the USDZ archive. This limitation will likely be addressed in a future version of USD. (See USD pull request #2133.)

USD Primvar data types

Blender supports a subset of the USD basic data types for import and export.

Only the types natively supported by Blender's attribute system will be processed.

Blender type	USD type	Notes
Boolean	bool	
8-Bit Integer	uchar	The USD unsigned 8-bit value will be cast to a signed value for import. The signed value will be cast to unsigned for export.
Integer	int	A 32-bit signed integer value.
Float	float	A 32-bit, single-precision, floating point value.
Vector	float3	3D vector with 32-bit floating-point values.
2D Vector	float2/texCoord2f	2D vector with 32-bit floating-point values.
Color	color4f	RGBA color with 32-bit floating-point values. As a special case, when encountering a Primvar or attribute for USD's `displayColor`, it will be read or written as `color3f` data with an Alpha component of 1.0.
Byte Color	color4f	USD does not provide a byte color equivalent. The byte values will be converted to float and exported as a color4f.
Quaternion	quatf	Floating point Quaternion rotation.

Implementation Caveats

Blender does not support USD Primvars using 64-bit integer values (`int64`), those using unsigned types (`uint`), or those using 64-bit double-precison or 16-bit half-precision floating-point values. For example, this would include such types as `matrix4d` (4x4 matrix of doubles) and `quath` (half-precision quaternion).

Note

The USD `float4` data type has no direct Blender equivalent and will not be treated as a Blender `Color` or `Quaternion`.