

An Empirical Analysis of Universal Scene Description (USD) Versus Traditional 3D Data Exchange Formats (FBX and Alembic): Integration, Extensibility, and Computational Efficiency in Digital Production Pipelines

Seok-Hyun Ahn ^{1,*}, Seung-hyun Lee ², Soon-chul Kwon ³, and Leehwan Hwang ⁴

¹ Graduate School of AI Policy and Strategy, Gwangju Institute of Science and Technology (GIST), Gwangju, Republic of Korea; elliott.ahn@gist.ac.kr

² Department of Immersive Convergence Content, Kwangwoon University, Seoul, Republic of Korea; shlee@kw.ac.kr

³ Department of Immersive Convergence Content, Kwangwoon University, Seoul, Republic of Korea; ksc0226@kw.ac.kr

⁴ Department of Immersive Convergence Content, Kwangwoon University, Seoul, Republic of Korea; optics.hwang@kw.ac.kr

* Correspondence

<https://doi.org/10.5392/IJoC.2026.22.2.030>

Manuscript Received 4 June 2025; Received 23 June 2026; Accepted 24 June 2026

Abstract: *This study compares Filmbox (FBX), Alembic, and Pixar's Universal Scene Description (USD) as data-exchange mechanisms for visual effects (VFX) and digital content creation pipelines. Beyond their roles as file specifications, the analysis differentiates their production functions: FBX for editable rigged assets and animation, Alembic for baked time-sampled geometry, and USD for composed scene description using layers, references, variants, and sparse overrides. For reproducibility, a controlled benchmark was conducted in Blender. Identical geometry and animation were exported to each format, and software versions, scene complexity, repetition counts, and timing boundaries were explicitly reported. The benchmark evaluated two update scenarios—an animation revision and a layout/camera change—and two distinct operations: a full DCC import and a native USD layer reload. Across both scenarios, the sparse USD update layer constituted only a small fraction of the base file, while direct FBX and Alembic updates necessitated republishing nearly complete files. Full referenced-stage imports were comparable to or faster than legacy-format update imports, and reloading only a changed layer in an already-open USD stage was significantly more efficient than any full import. These results affirm USD's primary advantage in sparse composition and non-destructive updates. However, they do not support a universal claim that every USD load path is faster. FBX and Alembic remain effective for their specialized handoff roles, while USD is better suited for large, evolving scenes and parallel departmental workflows. Detailed file-size and timing measurements are reported in Section 4.*

Keywords: Universal Scene Description (USD); Filmbox (FBX); Alembic; Visual Effects (VFX); Scene Composition

1. Introduction

In visual effects (VFX) production, complex three-dimensional data must be exchanged and preserved across specialized digital content creation (DCC) applications for modeling, animation, simulation, lighting, and rendering. Filmbox (FBX), a proprietary interchange format now maintained by Autodesk, is widely used to transfer models, skeletal hierarchies, skin weights, cameras, and animation between DCC applications and real-time engines [1]. Alembic, jointly developed by Industrial Light & Magic (ILM) and Sony Pictures Imageworks, is an open framework for application-independent storage of baked, time-sampled scene data, especially geometry caches [2]. These formats solved different handoff problems and continue to be valuable in production.

Universal Scene Description (USD), originally developed by Pixar, extends the exchange problem into scene composition. USD encodes hierarchical static and time-sampled data and provides composition operators such as references, sublayers, payloads, variants, and sparse overrides [3, 4]. These mechanisms allow a stage

to aggregate assets without flattening all contributing files and enable departments to author independent opinions over a shared scene. USD does not itself replace source control, asset management, or concurrency policy; instead, it supplies a scene-description model that those pipeline services can address consistently.

The scope of this paper is intentionally limited to FBX, Alembic, and USD. Other important formats serve different primary purposes. The Graphics Language Transmission Format (glTF) is specified as an API-neutral runtime asset-delivery format optimized for transmission and loading, whereas Extensible 3D (X3D) is an open international standard for publishing and interacting with 3D content on the Web [5, 6]. They are therefore not direct substitutes for the high-end VFX authoring handoffs examined here. FBX and Alembic were selected because they represent the two principal legacy patterns that USD most directly consolidates: editable rig/animation exchange and baked geometry-cache exchange.

This study contributes a workflow-oriented comparison of the three formats, a revised controlled benchmark based on files containing the same geometry and animation content, and a qualified analysis of loading, update storage, and reload behavior. Section 2 reviews the complementary roles of FBX and Alembic; Section 3 examines USD and the comparative capability model; Section 4 reports the benchmark environment, measurement procedure, results, and limitations; Section 5 discusses the interpretation and validity of the findings; and Section 6 summarizes the production implications.

2. Traditional Formats in the VFX Pipeline: FBX and Alembic

FBX and Alembic are commonly used together because they preserve different kinds of production information. FBX is selected when downstream users require editable transforms, rigs, skeletal animation, and scene objects. Alembic is selected when upstream evaluation should be removed and the result delivered as a stable geometry or transform cache. The following subsections describe their roles in animation, lighting, scene assembly, and asset revision.

2.1 Animation and Character Design

FBX can encapsulate a character's joint hierarchy, skin weights, blend shapes, and animation curves, allowing a receiving application to reconstruct an editable rigged asset. This makes it useful for transferring animation between tools, retargeting motion, or moving a character into a game engine [1], [7]. Fidelity still depends on the exporter, importer, constraint types, coordinate conventions, and the subset of the FBX software development kit implemented by each application.

Alembic does not preserve a procedural rig or the controls that generated a deformation. Instead, it records evaluated transforms and geometry samples over time. A character or simulation can therefore be delivered as a stable cache for downstream departments without requiring the originating rig or solver [2], [8]. FBX is consequently better suited to editable animation exchange, while Alembic is better suited to final or near-final evaluated results.

2.2 Lighting and Rendering

Lighting and rendering scenes often contain high-density animated geometry whose upstream rigs and simulation graphs are no longer needed. Alembic's cache-oriented design allows render and lighting applications to stream evaluated geometry and transforms without repeatedly solving the source animation [2], [8]. Cameras can also be exchanged, but renderer-specific shading networks and lights are not represented by a universal Alembic semantic model. FBX can carry cameras, lights, basic materials, and animation, yet it is generally less attractive than a geometry cache for dense simulation results.

2.3 Scene Assembly and Layout

Neither FBX nor Alembic was designed as a general-purpose, multi-file scene-composition system. FBX can store a hierarchy of objects in a single file, and Alembic can retain an object hierarchy within an archive, but neither provides USD's integrated model of references, payloads, sublayers, variants, and opinion strength. Traditional pipelines therefore tend to assemble scenes in a native DCC format and export a flattened FBX or Alembic result for the next department. This approach is predictable but duplicates data and makes local downstream edits difficult to preserve when an upstream file changes.

2.4 Asset Management and Revision

FBX and Alembic files normally represent specific published versions of an asset or cache. Naming conventions, databases, and source-control systems identify versions such as `tree_v002.fbx` or `character_cache_v005.abc`; the formats themselves do not provide a complete asset-resolution policy. Both can store custom metadata, but the meaning and discovery of that metadata are pipeline-specific. This external version-management model is workable for discrete handoffs, yet it becomes costly when many shots duplicate the same asset and each update requires replacement or re-import of a full file.

3. USD Overview and Comparative Capabilities

USD is a scene-description framework rather than only a serialization format. A USD stage presents a composed scene graph whose prims and properties may be contributed by multiple layers. References and payloads incorporate other assets, sublayers combine departmental opinions, variants encode selectable alternatives, and stronger layers can override weaker data without modifying the source [3, 4]. USD supports geometry, cameras, lights, time-sampled properties, skeletal data through `UsdSkel`, and shading descriptions through `UsdShade` or interoperable material schemas. Practical fidelity still depends on the schemas and features implemented by each application.

This composition model is the basis of USD's scalability and collaboration benefits. Layout, animation, effects, shading, and lighting can publish separate layers over a shared asset structure, allowing work to proceed in parallel while preserving provenance [9, 10]. Adoption nevertheless requires an asset resolver, naming rules, validation, schema governance, and tool support. USD should therefore be understood as an enabling framework whose benefits depend on pipeline architecture rather than as an automatic replacement for every FBX or Alembic handoff.

3.1 Data Interoperability and Scope

Table 1 separates FBX and Alembic rather than combining them into a single legacy category. This makes their complementary design goals explicit and provides a clearer basis for comparing them with USD.

Table 1. Comparison of FBX, Alembic, and USD capabilities in VFX pipelines.

Data or feature	FBX	Alembic	USD
Geometry	Meshes, curves, NURBS, transforms, and common scene objects; fidelity varies by implementation.	Meshes, subdivision surfaces, curves, points, transforms, and time-sampled caches; optimized for evaluated geometry.	Schema-based static and time-sampled geometry, instances, relationships, and extensible prim types.
Rigging and deformation	Supports skeletons, skin weights, blend shapes, and baked animation; application-specific rig logic may be lost.	Does not preserve procedural rigs; stores evaluated deformation or transform samples.	<code>UsdSkel</code> supports joints, skinning, blend shapes, and animation within a composed scene.
Animation	Keyframes and baked curves for supported properties.	Time samples for transforms and geometry; well suited to caches.	Time samples can be authored on most properties and composed across layers.
Materials and shading	Basic material parameters, assignments, and texture references; complex node graphs are not portable.	No universal shading-network model; face sets and custom properties may support external assignments.	<code>UsdShade</code> and MaterialX-oriented workflows can represent networks, but cross-application fidelity depends on shared schemas.
Lights and cameras	Supports cameras and common light types, subject to importer/exporter coverage.	Cameras are supported; lights lack a broadly portable standard schema.	Provides camera and light schemas plus renderer-specific extensions.

Hierarchy and assembly	Preserves a hierarchy within a file but lacks general multi-file composition.	Preserves archive hierarchy but primarily represents a cache snapshot.	Composed stage supports nested assets, references, payloads, sublayers, variants, and overrides.
Metadata	Custom properties and scene settings; semantics are application-dependent.	Arbitrary user properties; semantics are pipeline-dependent.	Structured metadata, custom namespaces, schemas, asset information, and resolver-addressable paths.
Non-destructive updates	Usually requires re-export, replacement, or a native DCC reference system.	Usually requires a new cache or an external stitching workflow.	Sparse stronger layers can override source data without rewriting the referenced asset.
Encoding and access	Proprietary format; binary and ASCII variants; commonly accessed through an SDK or application plug-in.	Open framework; Ogawa is the prevalent archive backend, with HDF5 retained in older workflows.	Open framework; human-readable USDA, binary USDC, and packaged USDZ encodings.

3.2 Cross-Compatibility and Format Selection

Cross-compatibility is not equivalent to semantic equivalence. FBX importers may interpret rig constraints, units, or materials differently; Alembic reliably conveys evaluated geometry but omits the procedural controls that produced it; and USD may preserve rich scene description that a particular DCC application does not fully consume. A production pipeline therefore needs a documented contract specifying supported schemas, units, naming, coordinate systems, materials, and validation rules. USD provides a broader container for that contract, but it cannot by itself guarantee identical results in every application [3], [11].

The format choice should follow the task. FBX remains appropriate for compact editable character and animation exchange; Alembic remains appropriate for stable high-volume caches; glTF and X3D are appropriate for runtime delivery and Web-oriented interactive publication; and USD is most valuable when assets must be assembled, revised, referenced, and shared across many departments. Hybrid pipelines are therefore not a transitional failure but often the technically correct allocation of formats to different stages.

3.3 Performance, Memory, and Scalability

Alembic is optimized for sequential input/output (I/O) of evaluated geometry and can provide excellent playback for a single cache. FBX performance depends strongly on the breadth of imported scene data and the host implementation. USD adds composition overhead, but references, instancing, population masks, and payloads can prevent unnecessary duplication and defer expensive data until it is needed [12]. Consequently, a small flat cache may load faster than a complex USD stage, while a large assembled scene may benefit substantially from USD's selective population and shared asset structure.

Memory behavior is similarly workload-dependent. Alembic readers can stream samples from a compact cache; FBX typically reconstructs editable application objects; and a USD stage retains composition information while sharing referenced and instanced data. Claims of universal superiority are therefore inappropriate. Performance must be measured for a specified host, scene, operation, cache state, and measurement boundary. Section 4 applies this requirement to a same-content benchmark and distinguishes full DCC import from native USD layer reload.

3.4 Workflow Efficiency and Flexibility

The principal workflow difference is that FBX and Alembic usually support linear publish-and-consume handoffs, whereas USD supports a composed set of opinions. This affects how artists make local changes, how departments work concurrently, and how a shot adopts an updated asset. The following mechanisms are central to that distinction.

3.4.1 Layering and Non-Destructive Editing

A downstream adjustment to an FBX or Alembic delivery often requires a new export or a copied file. In USD, an artist can author a stronger layer that changes visibility, transforms, materials, or other properties while leaving the referenced source untouched. The final value is resolved by composition strength, allowing the override to be enabled, replaced, or removed without destructively editing the asset [4], [9].

3.4.2 Variant Sets and Branching

Variant sets allow alternatives such as modeling levels, outfits, material looks, or prop states to coexist within an asset interface. A shot selects the required variant rather than managing unrelated copies. Variants are most effective when their purpose, nesting, and allowed combinations are governed by pipeline conventions; uncontrolled variant proliferation can otherwise increase composition and testing complexity.

3.4.3 Parallel, Non-Linear Workflows

Layout, animation, effects, shading, and lighting departments can publish separate USD layers over a shared stage instead of waiting for one flattened file to pass serially through the pipeline. This architecture reduces ownership conflicts at the file level and enables departments to receive upstream updates while retaining downstream opinions [9, 10]. Actual simultaneous editing still requires external locking, source control, review, and merge policies.

3.4.4 Version Management and Asset Updates

USD asset paths can be resolved through a pipeline-specific asset resolver, and metadata can identify asset names, versions, dependencies, and provenance. Updating a shot may therefore involve changing a reference or resolver context rather than replacing many local copies. With FBX and Alembic, version selection is generally encoded in file names or managed by an external database, after which each consumer reloads the published file. USD does not replace the external version system, but it provides a standardized composition point through which that system can update a scene [4], [13].

4. Controlled Reference-Based Benchmark

The original production-oriented test was replaced with a controlled benchmark so that the compared files, software versions, scene complexity, repetitions, and timing boundaries are explicit. The benchmark evaluates two update scenarios using one deterministic source scene and compares direct full-file delivery in FBX and Alembic with sparse reference-based delivery in USD.

4.1 Experimental Environment and Scene

Table 2. Experimental configuration and controlled scene.

Item	Specification
Software	Blender 5.1.2 (build ec6e62d40fa9), Python 3.13.5; built-in FBX, Alembic, and USD import/export operators [14-16].
Format versions	FBX binary 7.4; Alembic Ogawa archive generated by Blender 5.1.2; OpenUSD 25.8; binary USDC base files and ASCII USDA override layers.
Hardware and OS	Debian GNU/Linux 13 virtual environment; process pinned to four x86-64 virtual CPU cores; 4.0 GB RAM; GPU acceleration disabled; local container filesystem.
Scene complexity	48 mesh objects; 124,848 displayed vertices; one camera; 120 frames at 24 fps. The same geometry, transforms, animation, and camera content were exported to FBX, Alembic, and USD.
Scenario 1	Ten animated objects received revised transform samples over frames 56-65; the direct formats were republished as complete files, while USD stored a sparse stronger layer.
Scenario 2	Twenty layout objects and the camera pose/focal length were updated; the direct formats were republished as complete files, while USD stored a sparse stronger layer.

4.2 Equivalence Controls and Measurement Procedure

Renderer-specific materials and lights were excluded from the quantitative scene because Alembic does not provide a semantically equivalent, application-independent shading and lighting representation. This restriction ensures that the measured files contain the same representable geometry, transforms, animation, and camera content. After import, every FBX, Alembic, and USD case produced the same scene: 48 mesh objects and one camera, with 124,848 displayed vertices. Each referenced USD stage was also compared against a complete direct USD export at all changed attributes and time samples; no mismatches were detected.

Data redundancy was normalized within each format as defined in Equation (1):

$$R = (S_update / S_base) \times 100 \quad (1)$$

where S_update is the newly delivered file after a modification and S_base is the corresponding base file. This avoids misinterpreting differences in compression as duplication. For FBX and Alembic, S_update is a complete republished scene; for USD, S_update is the sparse USDA layer that references the unchanged USDC base. Storage reduction is $100 - R$.

Import time was measured with a monotonic nanosecond timer around the host import operator only; scene reset, process initialization, and result inspection were outside the timed interval. Files were read from local storage under a warm-cache interactive-use condition. One warm-up was discarded and three measured imports were summarized by the median and observed range. Native USD update time was measured separately by alternating two valid versions of the changed layer and invoking a forced reload on the already-open root layer. Two warm-ups were discarded and 30 reloads were measured. A native layer reload and a full DCC import are different operations, so they are reported separately rather than combined into a single improvement percentage.

4.3 Results

Table 3. File size and normalized update redundancy.

Scenario	Workflow	Base file (MB)	Update file (MB)	R (%)
Scenario 1: ten-frame animation update	Direct FBX	2.362	2.362	100.00
	Direct Alembic	2.229	2.229	100.00
	USD reference layer	1.725	0.0946	5.48
Scenario 2: layout and camera update	Direct FBX	2.362	2.362	100.00
	Direct Alembic	2.229	2.229	100.00
	USD reference layer	1.725	0.00356	0.21

$R = (\text{update-file size} / \text{same-format base-file size}) \times 100$. The corresponding USD storage reductions were 94.52% in Scenario 1 and 99.79% in Scenario 2.

Table 4. Loading and update timing results, reported as median seconds (range).

Operation	FBX	Alembic	USD
Initial full-scene import	0.273 (0.266-0.311)	0.414 (0.408-0.426)	0.257 (0.244-0.281)
Scenario 1 full update import	0.306 (0.302-0.313)	0.379 (0.378-0.409)	0.270 (0.263-0.272)
Scenario 2 full update import	0.285 (0.272-0.292)	0.419 (0.405-0.441)	0.246 (0.207-0.249)
Scenario 1 native layer reload	N/A	N/A	0.00138 (0.00134-0.00142)
Scenario 2 native layer reload	N/A	N/A	0.000456 (0.000400-0.000476)

Full imports: one warm-up plus three measured runs. Native USD reloads: two warm-ups plus 30 measured runs. N/A indicates that the direct FBX and Alembic protocol required a complete re-import rather than an equivalent sparse-layer reload.

The controlled file-size results show that direct FBX and Alembic updates reproduced essentially the entire base file, whereas the USD update layers stored only the changed opinions. The animation layer was 5.48% of the base USD file and the layout/camera layer was 0.21%, corresponding to storage reductions of 94.52% and 99.79% relative to a complete same-format duplicate. These values replace the earlier approximate redundancy claim with measured sizes and an explicit formula.

Initial import times were similar for FBX and USD in this Blender implementation, while Alembic was slower for the tested transform-animation scene. Based on the median values displayed in Table 4, the referenced USD stage was 11.8% and 13.7% faster than FBX in Scenarios 1 and 2, and 28.8% and 41.3% faster than Alembic, respectively. These results do not support a single 18% improvement value; the percentage depends on the baseline format and update scenario. When the host retained an already-open USD stage, reloading only the changed layer required 1.38 ms and 0.46 ms, demonstrating the advantage of sparse recomposition without reconstructing the complete DCC scene.

The distinction between full import and native reload is operationally important. A DCC application that converts USD into native objects on every import may realize only modest loading gains, whereas a USD-native

host that retains the stage can update a small layer with substantially less work. The benchmark therefore supports USD's compositional efficiency but not an unconditional claim that USD is always the fastest serialization format.

4.4 Limitations

The benchmark uses one synthetic scene, one DCC implementation, a virtualized CPU environment, warm local storage, and transform-dominant animation. Absolute times should not be generalized to Maya, Houdini, Katana, Unreal Engine, network storage, heavier deforming geometry, or renderer-specific materials. File size also measures serialization and duplication, not semantic fidelity. In addition, collaboration-conflict frequency and artist training time were not quantified because they require logged multi-user studies rather than a single-process benchmark. Future work should repeat the protocol on production assets, multiple DCC hosts, cold and network-cache conditions, and material and rigging workflows with explicitly defined fidelity criteria.

5. Discussion

The benchmark should be read as a test of a specific claim rather than as a global performance ranking. The relevant question is not whether USD is universally faster than FBX or Alembic, but whether USD's composition model changes the cost structure of common production operations. The measured results support that narrower and more defensible claim: when a scene is updated, a sparse USD layer records only the changed opinions and therefore occupies a small fraction of the base file, whereas a direct FBX or Alembic delivery republishes an essentially complete file. The advantage is structural—it follows from referencing and sparse overrides—rather than from faster serialization of an equivalent flat payload.

The distinction between a full DCC import and a native USD layer reload is central to interpreting the timing results, and conflating the two is the most likely source of overstated performance claims in the literature. A DCC application that converts USD into native scene objects on every import incurs costs comparable to importing any other format, so loading gains in that path are modest and host-dependent. By contrast, a USD-native host that retains an open stage can recompose only the changed layer, which is a fundamentally smaller operation. Reporting these two paths separately, as this study does, avoids attributing a layer-reload speed-up to full-scene loading and clarifies where USD's benefit actually arises.

These findings are consistent with the qualitative experience reported by studios that have adopted USD as an assembly and collaboration backbone, where the principal gains are attributed to non-destructive layering, parallel departmental work, and version-aware referencing rather than to raw file-loading speed [9-11]. The present contribution is to provide a controlled, same-content measurement so that storage and timing behavior can be examined under explicit conditions. At the same time, the controlled design constrains external validity: a single synthetic, transform-dominant scene on one virtualized host does not capture deforming geometry, renderer-specific materials, heavy shading networks, network or cold-cache storage, or the behavior of other DCC hosts. The benchmark therefore provides a controlled indication of USD's compositional advantage rather than a definitive cross-application performance figure.

Two practical implications follow. First, format selection should remain task-driven: FBX and Alembic continue to be appropriate for editable rig/animation handoffs and stable geometry caches, while USD is most valuable where assets are assembled, revised, referenced, and shared across departments. Second, realizing USD's measured advantages depends on pipeline architecture—asset resolution, schema governance, validation, and tooling that preserves an open stage—so the benefit is contingent on adoption decisions rather than guaranteed by the format alone.

6. Conclusion

FBX and Alembic remain effective because each solves a specific production problem. FBX transfers editable scene objects, rigs, and animation across many applications, while Alembic delivers stable evaluated geometry and transform caches. Their limitations arise when a production must assemble many assets, preserve local overrides, propagate versions, and allow departments to work over the same evolving scene without repeatedly flattening and duplicating data.

USD addresses these limitations through references, sublayers, variants, payloads, structured metadata, instancing, and non-destructive composition. The controlled benchmark found that sparse USD updates required only a small fraction of the base-file size, while direct FBX and Alembic updates required approximately complete copies. Full referenced-stage imports were faster than the tested FBX and Alembic

update imports, but the magnitude varied by scenario and baseline. The strongest performance benefit appeared when an already-open USD stage reloaded only the changed layer, confirming that USD's main advantage is workflow-level composition rather than guaranteed superiority in every flat-file operation.

A phased hybrid strategy is therefore appropriate. Studios can retain FBX for editable animation handoffs and Alembic for selected geometry caches while adopting USD as the assembly, referencing, and version-aware backbone. Successful adoption requires schema governance, asset resolution, validation, tool integration, and training. Broader multi-application benchmarks and logged collaborative studies are needed before generalizing the present measurements to all production environments.

Conflicts of Interest: The authors declare no conflict of interest.

Data and Code Availability: The experimental configuration and aggregate benchmark measurements are reported in Section 4 and Tables 3-4. No separate public data or code repository accompanies this submission.

Use of AI Tools: The authors used ChatGPT (OpenAI) for language editing and refinement of the manuscript, including the description of the experimental procedure. The authors reviewed, verified, edited, and approved all AI-assisted text and take full responsibility for the accuracy, integrity, and final content of the manuscript.

References

- [1] Autodesk, FBX | Adaptable File Formats for 3D Animation Software. Accessed: Jun. 21, 2026. [Online] Available: <https://www.autodesk.com/products/fbx/overview>
- [2] Alembic Project, Alembic Is an Open Framework for Storing and Sharing Scene Data, GitHub. Accessed: Jun. 21, 2026. [Online] Available: <https://github.com/alembic/alembic>
- [3] OpenUSD, USD Frequently Asked Questions. Accessed: Jun. 21, 2026. [Online] Available: <https://openusd.org/release/usdfaq.html>
- [4] OpenUSD, USD Terms and Concepts. Accessed: Jun. 21, 2026. [Online] Available: <https://openusd.org/release/glossary.html>
- [5] Khronos Group, glTF 2.0 Specification. Accessed: Jun. 21, 2026. [Online] Available: <https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html>
- [6] Web3D Consortium, What Is X3D?. Accessed: Jun. 21, 2026. [Online] Available: <https://www.web3d.org/x3d/what-x3d>
- [7] Foundry, Alembic and FBX. Accessed: Jun. 21, 2026. [Online] Available: https://learn.foundry.com/mari/content/user_guide/alembic_fbx/alembic_fbx.html
- [8] Autodesk, Alembic Caching - Maya Help. Accessed: Jun. 21, 2026. [Online] Available: <https://help.autodesk.com/view/MAYAUL/2024/ENU/?guid=GUID-9D272E39-9279-4146-8449-928DDA865C9D>
- [9] Pixar, Pixar's USD Pipeline: Make Complex Processes Predictable and Reliable. Accessed: Jun. 21, 2026. [Online] Available: <https://renderman.pixar.com/stories/pixars-usd-pipeline>
- [10] C. Dombrova, How Does USD Actually Get Used at a VFX Studio?, before & afters, May. 25, 2020. Accessed: Jun. 21, 2026. [Online] Available: <https://beforesandafters.com/2020/05/25/how-does-usd-actually-get-used-at-a-vfx-studio/>
- [11] Foundry, USD: What to Consider When Adopting Universal Scene Description. Accessed: Jun. 21, 2026. [Online] Available: <https://www.foundry.com/insights/film-tv/usd-things-to-consider>
- [12] OpenUSD, Maximizing USD Performance. Accessed: Jun. 21, 2026. [Online] Available: <https://openusd.org/release/maxperf.html>
- [13] OpenUSD, End-to-End Example. Accessed: Jun. 21, 2026. [Online] Available: https://openusd.org/dev/tut_end_to_end.html
- [14] Blender Foundation, FBX - Blender 5.1 Manual. Accessed: Jun. 21, 2026. [Online] Available: https://docs.blender.org/manual/en/latest/files/import_export/fbx.html
- [15] Blender Foundation, Alembic - Blender 5.1 Manual. Accessed: Jun. 21, 2026. [Online] Available: https://docs.blender.org/manual/en/latest/files/import_export/alembic.html
- [16] Blender Foundation, Universal Scene Description - Blender 5.1 Manual. Accessed: Jun. 21, 2026. [Online] Available: https://docs.blender.org/manual/en/latest/files/import_export/usd.html

