X3D and glTF Model Differencing for Conversions, Comparison and Conformance Testing

Rick Lentz

Naval Postgraduate School Monterey, California, USA, rlentz@gmail.com

Don Brutzman

Naval Postgraduate School Monterey, California, USA, brutzman@nps.edu

Michalis Kamburelis

Castle Game Engine and University of Wroclaw, Poland, michalis.kambi@gmail.com

The Extensible 3D (X3D) Graphics Architecture ISO/IEC 19775-1 International Standard version 4.0 is a major release that includes extensive support for version 2.0 of glTF (glTF2), a standard file format for 3D scenes, and models supporting geometry, appearance, a scene-graph hierarchical structure and model animation. X3D version 4 (X3D4) authors have the option to Inline (i.e. load) glTF models directly or else utilize native X3D nodes to create corresponding 3D models. Physically-based rendering (PBR) and non-photorealistic rendering (NPR), with corresponding lighting techniques, are each important additions to the X3D4 rendering model. These lighting models are compatible and can coexist in real-time with legacy X3D3 and VRML97 models (which utilize classic Phong shading and texturing) either independently or composed together into a single scene. The X3D4 approach to representing glTF2 characteristics is defined in complete detail in order to be functionally identical, thus having the potential to achieve the greatest possible interoperability of 3D models across the World Wide Web. Nevertheless, a persistent problem remains in that glTF renderers do not always appear to produce visually identical results. Best practices for mapping glTF structures to X3D4 nodes are emerging to facilitate consistent conversion. This paper describes a variety of techniques to help confirm rendering consistency of X3D4 players, both when loading glTF assets and when representing native X3D4 conversions. Development of an X3D4 conformance archive corresponding to the publicly available glTF examples archive is expected to reinforce the development of visually correct software renderers capable of identical X3D4 and glTF presentation.

CCS CONCEPTS • **differencing, model testing, spatial similarity testing**

Additional Keywords and Phrases: glTF, X3D

1 CAPABILITIES, PROBLEMS, AND OBJECTIVES

X3D4 is an International Standard used for various applications (including scientific visualization, medical applications, Web-based model display, and others). The X3D Architecture specification defines strictly measurable requirements for compliance without requiring specific software techniques for achieving them. Each technical capability is required to have at least two independent implementations (with at least one in open source) so that broad adoption is possible across a complete range of devices, hardware architectures, and programming languages. Multiple file-format encodings (XML, ClassicVRML, compressed binary, JSON) have equivalent expressive power for representing the X3D scene graph and event-animation graph. As a result, X3D models have the potential to run anywhere but are required to produce consistent renderings, animation, and interactive user experience (UX) to be considered compliant.

The KHRONOS group reports that "glTF™ is a royalty-free specification for the efficient transmission and loading of 3D scenes and models by engines and applications. glTF minimizes the size of 3D assets and the runtime processing needed to unpack and use them. glTF defines an extensible, publishing format that streamlines authoring workflows and interactive services by enabling the interoperable use of 3D content across the industry." Furthermore, "Khronos has created Tests and associated Process for the Specification to promote consistent multi-vendor implementations and to create an objective definition of conformance for the Specification..." [Khronos Group 2021] Conformance tests are only available to KHRONOS Adopters, and were not consulted as part of this work.

Comprehensive conformance testing is important when converting models between glTF and X3D [Kamburelis 2021]. Public glTF resources include over 75 model assets in a shared library, making consistency tests across a wide range of rendering requirements feasible. Both glTF and X3D4 specify unambiguous default values for every aspect of a renderable 3D model, so repeatable consistency testing appears to be feasible. Continuing cooperation between Web3D Consortium and Khronos Group will reduce differences and improve compatibility of implementations using both X3D and glTF [Web3D Consortium 2021].

This paper reviews various comparison methodologies when performing glTF to X3D conversions and comparisons in four renderers. Utilizing a variety of differencing ('diffing') techniques [McIlroy 1974] has proven exceedingly helpful when comparing and investigating inconsistent results. By automating the production of glTF asset conversions as X3D4 models using best practices for each, higher quality assurance (QA) between models, scenes, and renderings across all platforms becomes feasible.

2 PBR IN GLTF2 AND X3D4

Motivating capabilities are centered on establishing complete physically based rendering (PBR) and lighting, as defined in glTF2, consistently in X3D4. The X3D4 specification includes support for glTF inline rendering and ensures interoperable portability and use of standardized Physically Based Rendering (PBR) material extensions [Khronos Group 2021]. Additionally, real-time Web PBR rendering is an active research and development area, with frequent additional extensions.

The glTF2 repository [Khronos Group 2021] provides a diverse set of models with broad feature coverage useful for testing implementation coverage in X3D through rendering comparisons. Automating comparison of this diverse set of samples helps compare and contrast different X3D players for PBR consistency and other features over time (e.g., over each update).

Both glTF2 and X3D4 have a rich set of fundamental data structures for representing 3D geometry and appearance. Upgrading X3D4 to align with glTF is a major upgrade effecting many aspects of rendering and lighting. As an overview, additional and modified nodes in X3D4 Shape and Lighting components include:

1. Appearance node: added alphaCutoff and alphaMode fields, backMaterial, pointProperties.
2. Material node: retained all X3D3 material properties, added ambientTexture, diffuseTexture, emissiveTexture, normalTexture, occlustionTexture, shininessTexture and specularTexture. Also provided more precise mapping modes for texture coordinates relevant to each texture field.
3. PhysicalMaterial node: new Physically Based Rendering (PBR) capabilities matching glTF.
4. UnlitMaterial node: new simple Non-photorealistic rendering (NPR) capabilities matching glTF.

5. TwoSidedMaterial node: deprecated without loss of capability, better organization achieved.
6. Shape node: utility features for any geometry to display bounding box, cast shadows, or set geometry-related nodes as visible/nonvisible (similar to visibility of HTML elements).
7. Lighting component: added Unlit and Physical lighting modes to Phong and Gouraud models, added easily defined shadows model for all geometry.
8. EnvironmentLight node: support Image Based Lighting defined by glTF.
9. Texture projector component: enable perspective and parallel projection of textures as light sources, projected into a scene in a manner similar to other lighting sources.

Since more than one way can be found to render almost any 3D object, with varying fidelity of representation possible between perfect and approximate, best practices for conversion from glTF2 and X3D4 are not formally defined. Nevertheless, certain common data structures and modeling patterns are commonplace. Furthermore, variation considerations for consideration during conversion are well understood. It is therefore sensible that good techniques and best practices can be codified and improved, especially as well-defined common examples designed to test functional rendering features of glTF become closely scrutinized and compared using X3D. It is likely that both primary and alternative practices for mapping glTF2 assets into X3D4 representations will be usefully defined and compared.

3 MULTIPLE DIFFERENCING TECHNIQUES ADD VALUE

This submission version of your paper should not have headers or footers, these will be added when your manuscript is processed after acceptance. It should remain in a one-column format—please do not alter any of the styles or margins.

Measurable objectives for differencing techniques include geometry comparison, appearance comparison, animation comparison, technique refinement, error detection, and steady progress towards comparable consistency among multiple renderers. The following sections describe a series of comparison tests that together measure consistent achievement of these objectives.

3.1 TEXT AND STRUCTURED TEXT

When dealing with different data representations in the X3D4 specification, differencing tools provide rationales for understanding changes over time and between Web renders. The X3D Unified Object Model (X3DUOM) definitions match the formal requirements of the X3D Architecture in all respects and can autogenerate multiple representations, such as XML, ClassicVRML, JSON, and various programming languages (Java, Python, C, C++, and C#). Thus plain-text files using the XML X3D encoding have sufficient generality to stand in for any alternate X3D4 form.

Canonicalization (C14N) helps normalize X3D models as XML documents with strict syntax by removing default-value data, regularizing white space, and alphabetizing field information. As a result, files no longer include characters that are unnecessary for determining information differences. Comparison of unordered nodes in XML and JSON allows differencing at various levels in hierarchical data by first canonicalizing the files then comparing them (e.g. xmlint then diff). We perform canonicalization of models before saving a model or committing to version control. Current source code management systems (e.g., GitHub, Subversion version control) leverage complex diffing tools to provide accountability while tracking additions, deletions, and renaming over time. When appropriate, explicitly adding model metadata can help to further document variation rationales. Formal automated change management presents a clear picture of specific technical changes applied and how the difference impacts comparison of end results. Even so, more is needed to discern the impact of data changes on rendered results.

3.2 2D IMAGERY COMPARISONS

While direct comparisons between two images are relatively straightforward, multiple related issues pertain. Of note is that both X3D and glTF model parameters consistently include default values. Therefore expectations of consistent presentation are reasonable. For consistent image captures, values for appearance, materials, camera position, and lighting must all be consistent. Initial objectives include diffing between versions to identify visual similarity and dissimilarity. Figure 1 provides an example.

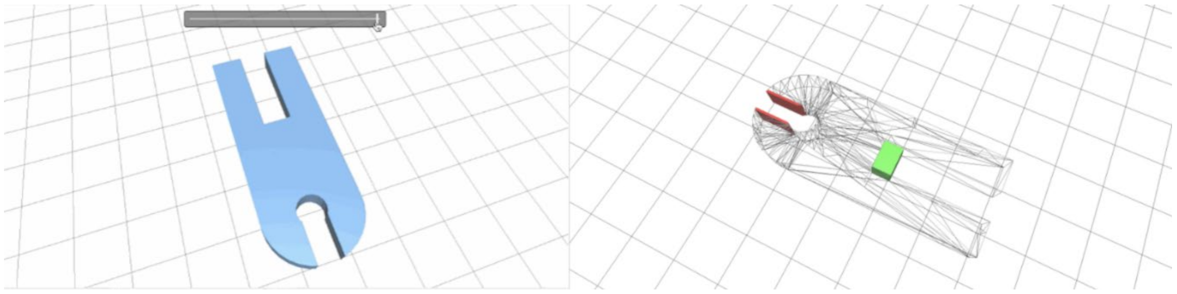


Figure 1: Examples of visual differencing to highlight changes in geometry and rendering [GitHub 2021].

Quantitative 2D metrics for image differencing are useful. The Google ModelViewer Render Fidelity Comparison Results project reports that "The purpose of glTF is to standardize Physically-Based Rendering (PBR) materials such that you can be confident your model will appear as intended in any lighting environment in any renderer." [Google 2021] Numeric differencing metrics are provided. For example, Figure 2 by the ModelViewer team shows a prominent glTF example that appears excellent but is not always consistently rendered.

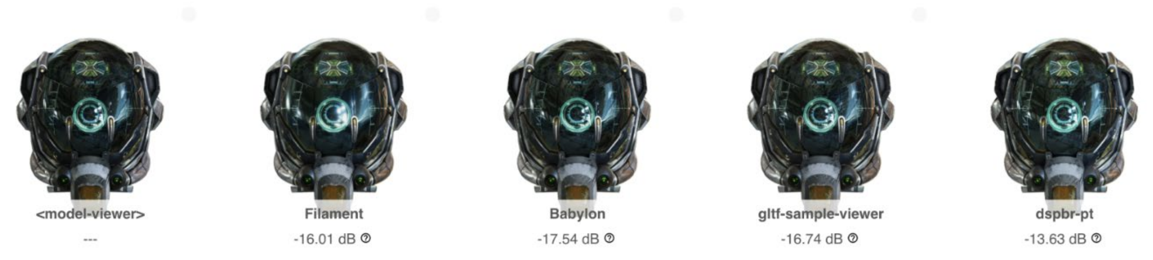


Figure 2: Strict Comparison of Damaged Helmet glTF model [Khronos Group 2021] example using multiple glTF viewers [Google 2021]

X3D4 further defines default values for various node attributes that can proactively account for sources of variability and encourage a high degree of cross-standard portability. For example, the X3D Viewpoint node contains default values for the center of rotation, orientation, position, and field of view to account for the camera defaults unambiguously. Similar constructs are available in glTF. Overall default rendering settings can be influenced by the host's screen size, aspect ratio, and hardware capabilities. While these default behaviors are independent of author control, but can either be specified on a per-player basis or else in allowed size as part of an HTML page. Other height-by-width size and aspect-ratio assumptions help constrain the rendered image variations to minimize image-difference comparisons. These steps can be repeated across a set of calibrated model viewpoints.

The following images show a left-to-right side-by-side comparison of the Damaged Helmet sample rendered as glTF in the glTF Sample Viewer, then rendered as converted X3D in view3Dscene, X3DOM, and X_ITE viewers. While the geometry mesh (e.g. IndexedTriangleSet) and camera perspectives are similar, differences in render support for Physical Materials, background defaults, and scene lighting defaults account for differences in the 2D renderings. Future conformance testing will likely automate the creation of 2D snapshots of both inline glTF and converted glTF in X3D players.

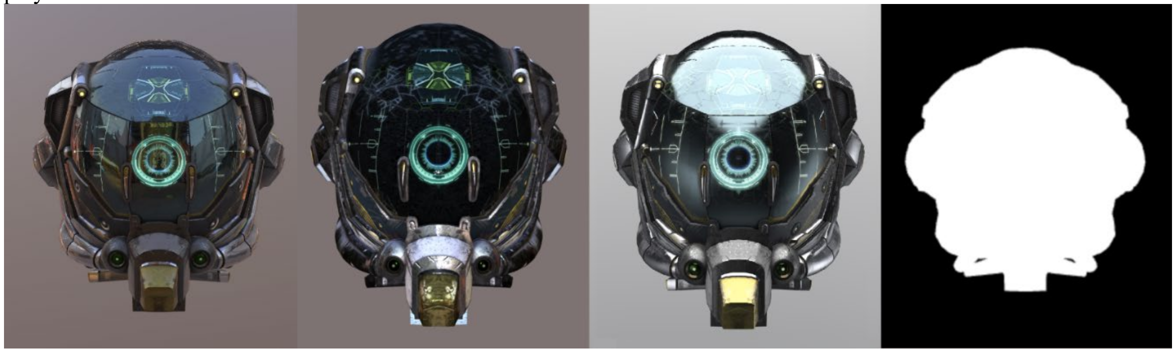


Figure 3: Comparison of a single model rendered (a) natively as glTF in glTF Sample Viewer, then converted to X3D in (b) view3dscene, (c) X3DOM, and (d) X_ITE players.

Viewpoints need to be centered appropriately so that each model is consistently visible. Inspection of current online example models reveals that few include any information about model size, model center, or preferred viewpoint. This gap indicates that conformance-testing conventions need to be defined (and perhaps automated) to provide Viewpoints that center each model at appropriate scaling distance within a well-defined aspect ratio and viewing frame. Image differencing provides benefits for each software change, historically in version control, and in unit testing prior to software release.

3.3 3D MODEL GEOMETRY

Having an efficient data representation for 3D objects that compose scenes is an important area of research. Data efficiency methods such as color quantizing of significant digits to not exceed the limits of human perception, impact data storage encodings and, therefore, interoperability comparisons. It is worth covering how the X3D specification leverages these. X3D4 supports efficient binary encoding through two paths, X3D CBE [Web3D Consortium 2015] or Compressed Binary Encoding (19776-3) and X3D EBE or Efficient Binary Encoding (19776-4). CBE leverages Fast Infoset (FI) Compression (ISO 24824-1) while EBE compresses using Efficient XML Interchange Compression [Brutzman 2015].

Through interoperability, X3D can leverage data efficiency methods within VRML and glTF2. For information compression, VRML models can use the Chisel library [Louka 2010] to clean (e.g., remove default fields), condense, reduce, and reorganize data. For geometry compression, glTF2 leverages Draco, an open-source library for compressing and decompressing 3D geometric meshes and point clouds intended to improve the storage and transmission of 3D graphics [Galligan 2021].

Comparing 3D meshes can be challenging for several reasons. First, unordered elements can have the same overall shape. Other challenges are when composing the same shape using different quantities of polygons. Multiple-sided polygons (N-gon representations) can be nonplanar and yield differing quantities of polygons to approximate surface shape with varying representation lengths, useful when downsampling objects in complex scenes. Each such situation is aided by differencing methods or similarity measures (e.g., Chamfer loss calculation or the logarithmic average of the pixel differences, excluding the transparent background).

Progressive geometric reduction of mesh shapes can occur via iterative resampling (mesh refinement) or applying curve-fitting techniques to produce parametric surfaces. Mesh refinement is the adjustment of vertices and faces to provide a pleasing geometry that reduces the data size. Parametric surfaces such as extrusions and Non-Uniform Rational B-spline Surfaces (NURBS) provide a procedural encoding of actions that describe smooth geometric shapes given just a few functional input parameters.

Perceptual differences are perhaps the most significant metrics for such variations. Visual comparison of screenshot captures corresponding to model viewpoints can verify whether per-pixel similarity is achieved. Recent work on appearance-driven differential rendering shows that perceptual differences can have a big impact on compression techniques [Nicolet 2021, Hasselgren 2021]. Of further interest is that the same kinds of comparison techniques described in this paper are essential for assessing model improvements in those references.

3.4 VIDEO PLAYBACK AND FUNCTIONAL TESTS FOR CONFIRMING 3D ANIMATION AND USER INTERACTION

Video playback can capture variations in spatial perspectives and scene dynamics, such as replaying a set of objects' positions and state changes over time (interpolator replays). When differentiating between video renderings, the comparison process is performed step-by-step at the frame level accounting for the same sources of divergence introduced in the 2D Imagery section. The following method is applied to account for default differences between renders:

1. Camera position adjustments are made to account for differences in default camera positions against the reference rendering.
10. Default lighting node differences are adjusted, again towards the reference.
11. Shader or physically based material nodes are adjusted to minimize the loss with each frame.

When comparing animations using stream capture and distillation, two types of comparisons are common: keyframe animation, which is time-based and driven by interpolators, or else selected animation such as a scene or viewpoint change. Sequential comparison of multiple keyframe 2D snapshots can then reveal whether variations exist.

For HTML-based presentation and interaction, Selenium provides a library and control system for automating the testing of Web applications, including Web-renderers [Software Freedom Conservancy 2021]. The ability to capture browser renderings for a set of scenes and animations and compare them over time through a simple script makes it a valuable tool for comparison analysis.

4 EXPERIMENTAL APPROACH

Modern 3D graphics is mathematical and deterministic, yet results often depend on arbitrary hidden factors embedded in software tools. In order for two standards to be aligned satisfactorily, an experimental approach is necessary to demonstrate correct achievement of goal requirements across diverse suites of tools serving many application domains.

Implementation, evaluation, and spiral improvement require an ongoing refinement process. We describe three tangible activities that permit measurement, evaluation, and improvement.

4.1 X3D MULTITEXTURE IMPLEMENTATION

As a preparation activity, we revisited visual inspection within a single scene using multiple objects, each expressing different attributes of the X3D specification. For example, the MultiTextureTeapot model places multiple instances of the same geometric object to test the implementation of different TextureCoordinateGenerator modes within the X3D specification. Although the MultiTexture node has been well defined since X3D version 3.1, support is currently inconsistent across existing players. MultiTexture node capabilities are now fully integrated with the X3D4 node definitions supporting glTF. The following example can be used to improve conformance.

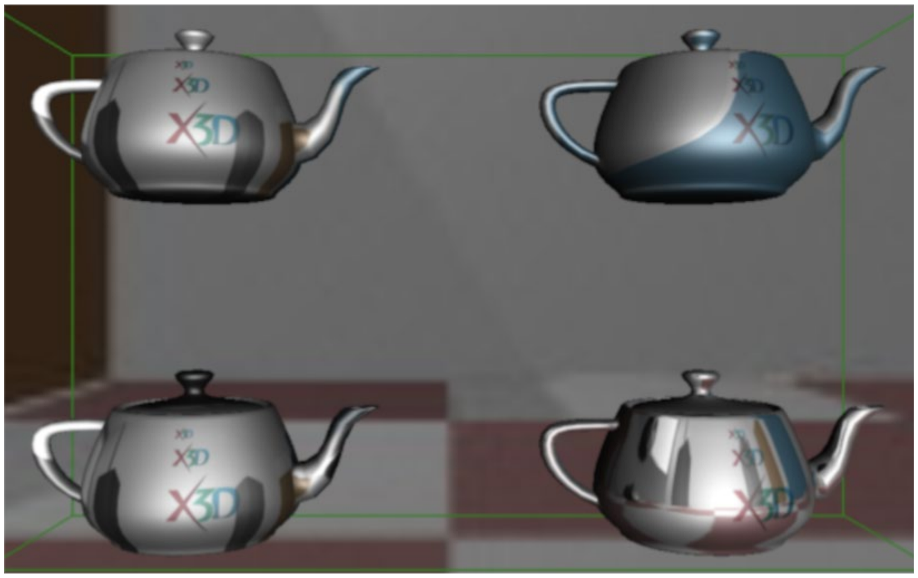


Figure 4: MultiTextureTeapot.x3d Showing Different TextureCoordinateGenerator Modes

4.2 CONVERSION OF GLTF EXAMPLES INTO X3D

Current experiments start by gathering the glTF2 examples from Khronos Group glTF-Sample-Models repository on GitHub [Khronos Group 2021]. The script identifies the path of all glTF samples in the directory and then iteratively uses view3Dscene [Kamburelis 2021] to perform glTF2 conversion to X3D4 [Web3D Consortium 2021]. The view3Dscene player is also used to generate reference screenshots of each sample model.

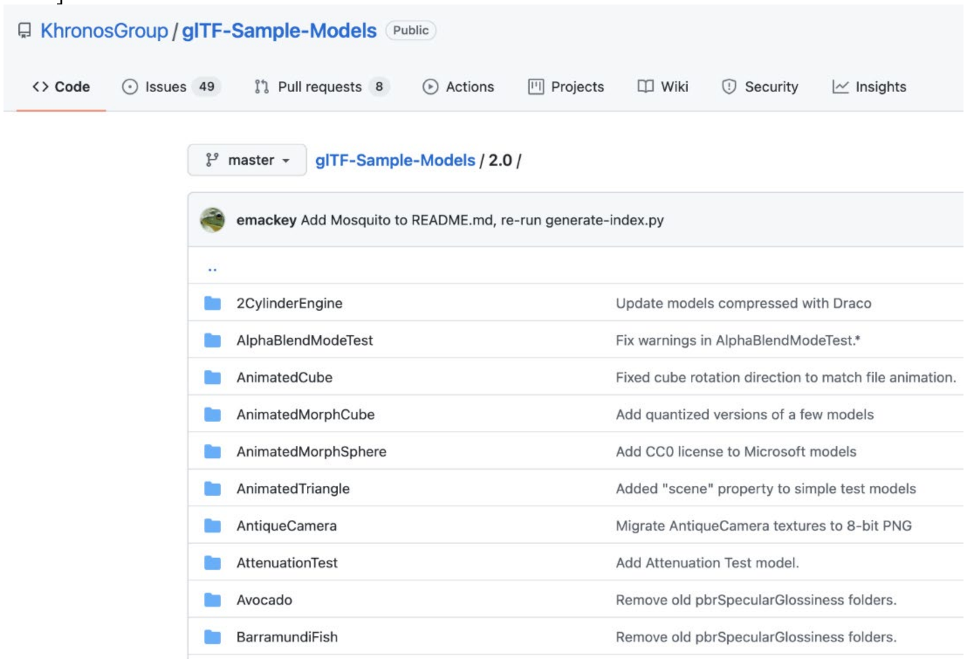
Given a set of X3D4 models matching glTF2 sample assets, the next step in this process is to prepare the HTML from existing examples for various viewers, including X_ITE [Seelig 2021] and x3dom [Plesch 2021]. Further automation using open-source X3DOM and X_ITE implementations is similarly feasible. The base HTML pages are essentially the same for all examples, and only the local X3D4 converted samples change.

A local HTTP service serves up the generated HTML, associated X3D files, and mitigates Cross-Origin Resource Sharing (CORS) issues associated with local assembly of assets, allowing local testing and author improvements to occur prior to publication on a server. Selenium can programmatically capture both X_ITE and X3DOM renderings for each example model. A series of Ant build tasks can create comparison images for each of these 76 current glTF2 sample scenes. All files and corresponding comparison renderings are checked into GitHub so that anyone can leverage GitHub's built-in image comparison tools 2-up, Swipe, and Onion Skin [GitHub 2021] to publicly browse differences. Diff histories are already available, checking reference images into version control can further demonstrate consistency (and variations) over time.

Image differencing tools can identify model differences for (a) player loading the original glTF or presenting the converted X3D, (b) for regression testing of players as their software evolves, and (c) for comparison testing between players to show conformance. Keyframe animation can be similarly checked via multiple screenshots, taken either at interpolator changes or at periodic intervals.

4.3 ONLINE EXAMPLE ARCHIVES

The glTF2 archive contains 76 models, contributed to provide coverage to specific elements of the specification and extensions. Additionally, the repository links to several other glTF repositories, e.g., the Smithsonian's 3D Open Access Repository that contains digital scans, some of which are several hundred megabytes. All Blender models can also be a testing source for glTF models. Thus a very wide range of glTF models are available for use in (or export to) X3D4 [Brutzman 2021].



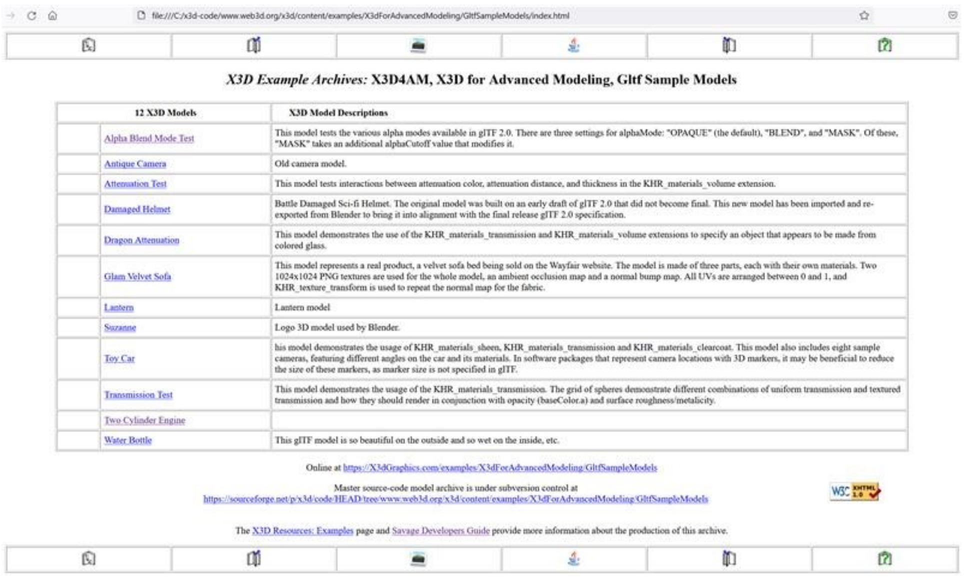
emackey Add Mosquito to README.md, re-run generate-index.py

..

2CylinderEngine	Update models compressed with Draco
AlphaBlendModeTest	Fix warnings in AlphaBlendModeTest.*
AnimatedCube	Fixed cube rotation direction to match file animation.
AnimatedMorphCube	Add quantized versions of a few models
AnimatedMorphSphere	Add CC0 license to Microsoft models
AnimatedTriangle	Added "scene" property to simple test models
AntiqueCamera	Migrate AntiqueCamera textures to 8-bit PNG
AttenuationTest	Add Attenuation Test model.
Avocado	Remove old pbrSpecularGlossiness folders.
BarramundiFish	Remove old pbrSpecularGlossiness folders.

Figure 5: glTF2 Sample Model repository [6] covers technical features in glTF2

The X3D for Advanced Modeling Example Archive currently contains a dozen examples detailing various capabilities of the X3D4 specification for glTF as shown in Figure 6. These examples are often of benefit for narrowing in on specific details needed to test relevant use cases.



file:///C:/x3d-code/www.web3d.org/x3d/content/examples/X3dForAdvancedModeling/GltfSampleModels/index.html

X3D Example Archives: **X3D4AM, X3D for Advanced Modeling, Gltf Sample Models**

12 X3D Models	X3D Model Descriptions
Alpha Blend Mode Test	This model tests the various alpha modes available in glTF 2.0. There are three settings for alphaMode: "OPAQUE" (the default), "BLEND", and "MASK". Of these, "MASK" takes an additional alphaCutoff value that modifies it.
Antique Camera	Old camera model.
Attenuation Test	This model tests interactions between attenuation color, attenuation distance, and thickness in the KHR_materials_volume extension.
Damaged Helmet	Battle Damaged Sci-fi Helmet. The original model was built on an early draft of glTF 2.0 that did not become final. This new model has been imported and re-exported from Blender to bring it into alignment with the final release glTF 2.0 specification.
Dragon Attenuation	This model demonstrates the use of the KHR_materials_transmission and KHR_materials_volume extensions to specify an object that appears to be made from colored glass.
Glam Velvet Sofa	This model represents a real product, a velvet sofa bed being sold on the Wayfair website. The model is made of three parts, each with their own materials. Two 1024x1024 PNG textures are used for the whole model, an ambient occlusion map and a normal bump map. All UVs are arranged between 0 and 1, and KHR_texture_transform is used to repeat the normal map for the fabric.
Lantern	Lantern model
Suzanne	Logo 3D model used by Blender.
Toy Car	his model demonstrates the usage of KHR_materials_sheen, KHR_materials_transmission and KHR_materials_clearcoat. This model also includes eight sample cameras, featuring different angles on the car and its materials. In software packages that represent camera locations with 3D markers, it may be beneficial to reduce the size of these markers, as marker size is not specified in glTF.
Transmission Test	This model demonstrates the usage of the KHR_materials_transmission. The grid of spheres demonstrate different combinations of uniform transmission and textured transmission and how they should render in conjunction with opacity (baseColor.a) and surface roughness/metalicity.
Two Cylinder Engine	
Water Bottle	This glTF model is so beautiful on the outside and so wet on the inside, etc.

Online at https://X3dGraphics.com/examples/X3dForAdvancedModeling/GltfSampleModels

Master source-code model archive is under subversion control at
https://sourceforge.net/p/x3d/code/HEAD/tree/www.web3d.org/x3d/content/examples/X3dForAdvancedModeling/GltfSampleModels

The X3D Resources: Examples page and Savage Developers Guide provide more information about the production of this archive.

Figure 6: X3D for Advanced Modeling (X3D4AM) Example Archive includes converted X3D versions of glTF sample models [Web3D Consortium 2015]

Further optimizations are needed, and further exploration is warranted. All project source code and comparison scripts are available under open-source licenses. All comparison examples are being prepared for eventual viewing, exploration, and X3D4 conformance testing as part of the X3D Example Scene Archives [Brutzman. 2021]. Producing a glTF2-X3D4 conformance suite of automated repeatable tests is expected to yield numerous benefits.

5 CONCLUSIONS

The addition of glTF2 capabilities to X3D is a major advance that upgrades a lighting and rendering model which has remained stable for two decades. Meanwhile, advanced PBR often suffers from inconsistent rendering and presentation. The entire purpose of this paper is to demonstrate that the advanced capabilities are indeed compatible with the classic Phong rendering model and that conformant, consistent presentation as required by the X3D International Standard is also achievable. In the conduct of this work, we have determined that advanced differencing and testing techniques might broadly improve quality for all users on the Web.

This paper lists multiple differencing methods to test glTF2 support across X3D4 implementations and review the changes in the X3D specification to enable glTF2 interoperability. Advanced rendering techniques are now stabilized via the glTF2 standard, but few implementations produce visually identical results when rendering the same glTF2 model asset. Adding strict defaults for all fields in X3D4 and becoming deliberate about model comparison can eventually isolate and resolve unintended implementation differences.

6 RECOMMENDATIONS FOR FUTURE WORK

Eventually contributing a repeatable online conformance repository with automated scripts for testing and viewing results in precise interoperability between renderers. Such consistency can further confirm that all technical aspects of both the glTF2 and X3D4 standards are well specified. Several enhancements and directions for future work become possible. One enhancement can enable glTF2 to X3D4 conversions to undergo conformance checks [Brutzman 2021]. Mesh refinement through information reduction, geometric reduction, or parametric-surface approximation can be quantitatively tested. Given increasing interest in performing point cloud distillation into geometric approximations, similar techniques pertain. Topics such as remeshing, color quantization, and other simplification methods are part of model compression and will be revisited when the X3D compressed binary encoding is updated. The Humanoid Animation (HAnim) International Standard needs to be compared to corresponding glTF work. Shape recognition techniques are likely to become increasingly valuable since precise representation of underlying geometric structure can assist effective optimization.

For nearly every conformance question, a variety of tools and techniques offer answers to the classic question "how do you measure that?" Connecting and instrumenting conversion and comparison processes for X3D and glTF models can provide significant benefit to all publishers and users of 3D graphics on the Web.

ACKNOWLEDGMENTS

The authors thank all glTF model donors plus all contributors to the glTF 4.0 and X3D 2.0 specifications.

7 HISTORY DATES

Received August 2021; revised October 2021; accepted TBD

REFERENCES

Khronos Group. glTF Statement of Purpose. Retrieved Sept 28, 2021 from https://www.khronos.org/gltf

Michalis Kamburelis. 2021. Converting from glTF 2.0 to X3D 4.0. Retrieved on Oct 3, 2021 from https://github.com/michaliskambi/x3d-tests/wiki/Converting-glTF-to-X3D

Web3D Consortium. 2021. Web3D Consortium and Khronos Group deepen cooperation on open standards for 3D on the Web. Accessed October 2nd, 2021 from https://www.web3d.org/news-story/web3d-consortium-and-khronos-group-deepen-cooperation-open-standards-3d-web

Douglas McIlroy. Jun 15, 1974. Initial developer of diff open-source application while at AT&T Bell Laboratories.

Khronos Group. 2021. glTF online viewer for PBR samples. Retrieved July 17, 2021 from https://github.khronos.org/glTF-Sample-Viewer-Release/

Khronos Group. 2021. glTF 2.0 sample repository. Retrieved July 11, 2021 from https://github.com/KhronosGroup/glTF-Sample-Models/tree/master/2.0

GitHub, Inc. 2021. GitHub built-in image comparison tools. Retrieved Aug 8, 2021 from https://docs.github.com/en/github/managing-files-in-a-repository/working-with-non-code-files/rendering-and-diffing-images

Google Inc. 2020. ModelViewer comparison methodology. Retrieved Sept 27, 2021 from https://modelviewer.dev/fidelity/

Web3D Consortium. 2015. X3D Compressed Binary Encoding Activity 2015. Accessed Aug 7, 2021 from https://www.web3d.org/working-groups/x3d/compressed-binary-encoding-activity

Don Brutzman. 2015. X3D Efficient Binary Encoding Progress Summary. Accessed Aug 7, 2021 from https://www.web3d.org/sites/default/files/page/X3D%20Compressed%20Binary%20Encoding%20Activity/X3dEfficientBinaryEncodingQuicklookSummary2015August23.pdf

Michael Louka. 2010. Chisel 2.1.4 (HVRC Edition) VRML Optimizer. Accessed Aug 7, 2021 from https://www2.hrp.no/vr/tools/chisel/index.html

Frank Galligan. 2021. Draco Bitstream Specification. Accessed Aug 7, 2021 from https://google.github.io/draco/spec/

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large Steps in Inverse Rendering of Geometry. ACM SIGGRAPH Asia 2021 Retrieved Oct 1, 2021 from https://rgl.s3.eu-central-1.amazonaws.com/media/papers/Nicolet2021Large_4.pdf

Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala and Samuli Laine. 2021. Appearance-Driven Automatic 3D Model Simplification. Retrieved May 22, 2021 from https://research.nvidia.com/publication/2021-04_Appearance-Driven-Automatic-3D

Software Freedom Conservancy. 2021. Selenium driver for Chrome. Retrieved July 10, 2021 from https://github.com/SeleniumHQ/selenium

Michalis Kamburelis. 2021. View 3D Scene application. Retrieved May 22, 2021 from https://github.com/castle-engine/view3dscene

Web3D Consortium. 2021. X3D4 Architecture Committee Draft (WD3) Specification uses HTML/CSS diffs to show primary functional differences between X3D3 and X3D4. Published December 2020. https://www.web3d.org/specifications/X3Dv4Draft/ISO-IEC19775-1v4-WD3

Holger Seelig et. al. 2021. X_ITE X3D Browser. Retrieved June 5, 2021 from https://github.com/create3000/x_ite

Andreas Plesch et. al. 2021. X3DOM. A framework for integrating and manipulating X3D scenes as HTML5/DOM elements. Retrieved June 6, 2021 from https://www.x3dom.org/

Web3D Consortium. 2015. X3Dv4 Public Working Draft. Retrieved May 15, 2021 from https://www.web3d.org/x3dv4-public-working-draft

Many model source references https://opengameart.org, https://kenney.nl, https://sketchfab.com/features/gltf, https://quaternius.com, https://blendswap.com

This work's archive repositories https://x3dgraphics.com/examples/X3dForAdvancedModeling/#GltfSampleModels

Don Brutzman. 2021. X3D Scene Archives. Accessed May 8, 2021 from https://www.web3d.org/x3d/content/examples/X3dResources.html#Examples

Don Brutzman. 2021. X3D Edit Verion 4.0. Retrieved Jun 20, 2021 from https://savage.nps.edu/X3D-Edit/#Downloads