

Web3D 2026

13-15 October 2026

Doha, Qatar

Implementing X3DCPPSAIL with AI-Assistance

An ISO/IEC 19777-4 C++ Scene Authoring Interface library for the X3D 4.1 node set

Ryker Ellingworth
University of North Dakota
School of Electrical Engineering & Computer Science

Aaron Bergstrom
University of North Dakota
Computational Research Center, DREAM Lab

Don Brutzman
Web3D Consortium



Web3D 2026

The Question Behind This Project

What does it look like to build a SAI library with AI assistance, and where does that collaboration help most?

PHASE 1

AI-assisted learning and development

PHASE 2

Deterministic generation with X3DUOM and XSLT

How Code Meets X3D

ISO/IEC 19775-1

Architecture

Defines X3D scenes: nodes, fields and components.

ISO/IEC 19775-2

Scene Authoring Interface

How applications inspect and manipulate the scene graph.

ISO/IEC 19777-1…7

Language bindings

Map the SAI to ECMAScript, Java, C, C++, C#, Python and Pascal.

The gap: the C++ binding, ISO/IEC 19777-4, only covered the X3D v3.3 node set.

Why C++, and Why X3D 4.1

WHY C++

The language of 3D engines

- One of the most popular languages for engines and tooling
- A native, typed scene graph removes a translation layer
- Adobe’s USD file-format plugins are written in C++

WHY X3D 4.1

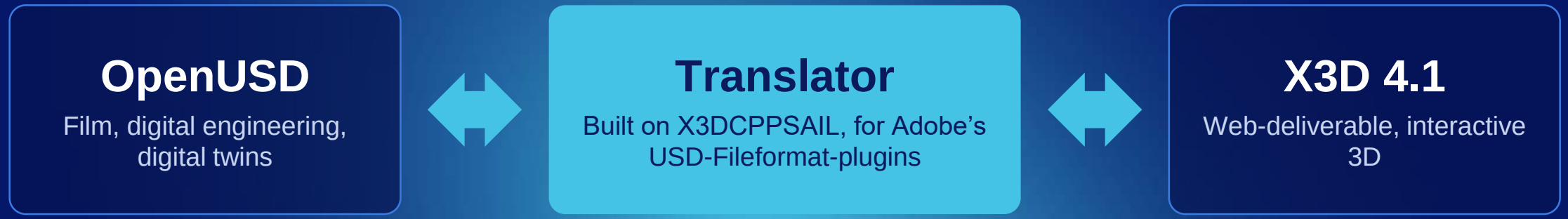
Target the draft, not 4.0

- Committee Draft, a strict superset of 4.0
- New rendering, lighting and geometry nodes
- Aligns far better with OpenUSD

`EnvironmentLight` `Tangent` `GaussianSplats`



The Goal: An OpenUSD ⇄ X3D Translator



OPENUSD

Pixar's in-house format, now an open standard; AOUSD ratified Core Specification 1.0 on Dec 17, 2025.

UND DREAM LAB

A member of both the Web3D Consortium and AOUSD, building the bridge between them.

Building on Lee’s C++ Binding

Rather than design a new binding, X3DCPPSAIL keeps Myeong Won Lee’s ISO/IEC 19777-4 architecture and updates the node set to X3D 4.1.

A second C++ SAI, Ben Sandbrook’s x3d-cpp (Virginia Tech), was learned of later and was not examined for this work.

CARRIED OVER

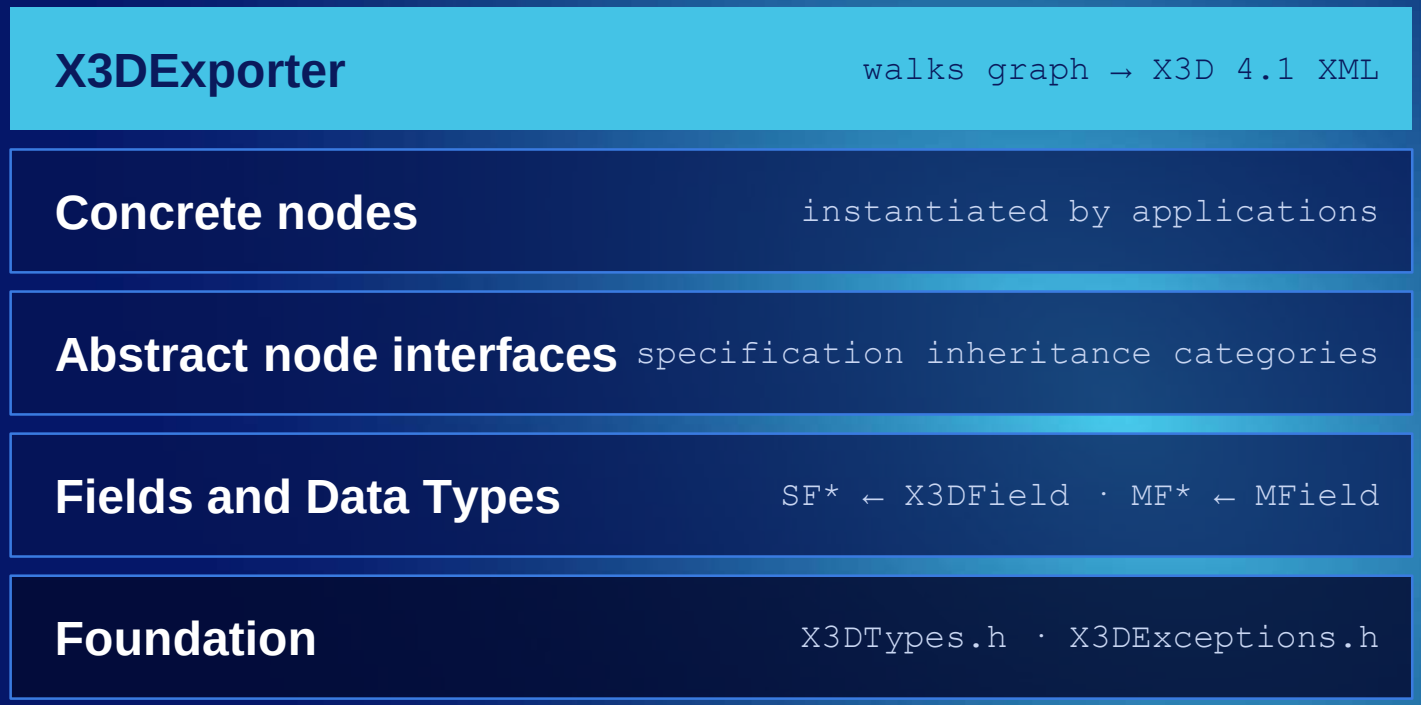
- Primitive type definitions
- Abstract node interfaces
- Concrete node classes
- Get/set field accessors

THE ONE EXCEPTION

Nodes deriving from more than one abstract type, such as ImageTexture



A Layered Static Library



BUILD

Visual Studio, C++17

VERIFICATION

A separate tests console app

SHARED DEFAULTS

X3DField supplies isReadable() and isWritable() once

PART TWO

Where the AI Assistant Helped

Claude (Opus 4.8) was used throughout development, and we studied its output as we went.



Four Roles the Assistant Played

1

Specification reader

Digested the cross-referenced 3.3 binding and 4.1 draft so coding could begin.

2

Code generator

Drafted node classes from one template; the team reviewed, fixed and tested.

3

Tutor

Explained choices like static_cast vs dynamic_cast, and when virtual is needed.

4

Design checker

Chose among options the programmer laid out, e.g. grouping metadata nodes.

Generate, Review, Fix, Test

01

Generate

Claude drafts a group of node classes

02

Review

The team checks for errors

03

Fix

Problems corrected by hand

04

Test

Next group waits until this one passes

IN-EDITOR · VISUAL STUDIO

Flags errors, suggests fixes and small design tweaks

FULL CHAT

Big jobs: fetching and reasoning over whole specifications

Strong on Breadth, Shaky on Precision

Strengths

BREADTH TASKS

- Generating repetitive structure
- Explaining unfamiliar concepts
- Combing through large specifications
- Moved faster than otherwise possible

Weaknesses

PRECISION TASKS

- Occasional X3D specification hallucinations
- Version details needed manual checks
- Design problems caught only by review
- Extra manual review slowed things

We did not measure whether review time exceeded the time saved.

x3d_mcp was not used for the duration of this project.



Two Phases of One Method

PHASE 1

AI-assisted exploration

No 4.1 C++ binding existed, so AI helped discover the patterns.

PHASE 2

Deterministic generation

X3DUOM and XSLT turn those patterns into the full binding.

X3D Schema
annotated XML Schema

X3DUOM
unified object model

XSLT
stylesheets

C++ binding
X3DCPPSAIL

The same pipeline already maintains the Java and Python SAI libraries and the X3D Ontology.

Deterministic generation is a work in progress.

Regression Testing at Archive Scale

4,000+

models in the X3D Examples Archives

- Each model already published as XML, VRML, JSON, Turtle, Java and Python
- A new X3dToCpp.xslt stylesheet will add C++ to the model conversion suite
- Thousands of real scenes to check coverage and correctness



Conclusions and Next Steps

COMPLETE AND TESTED

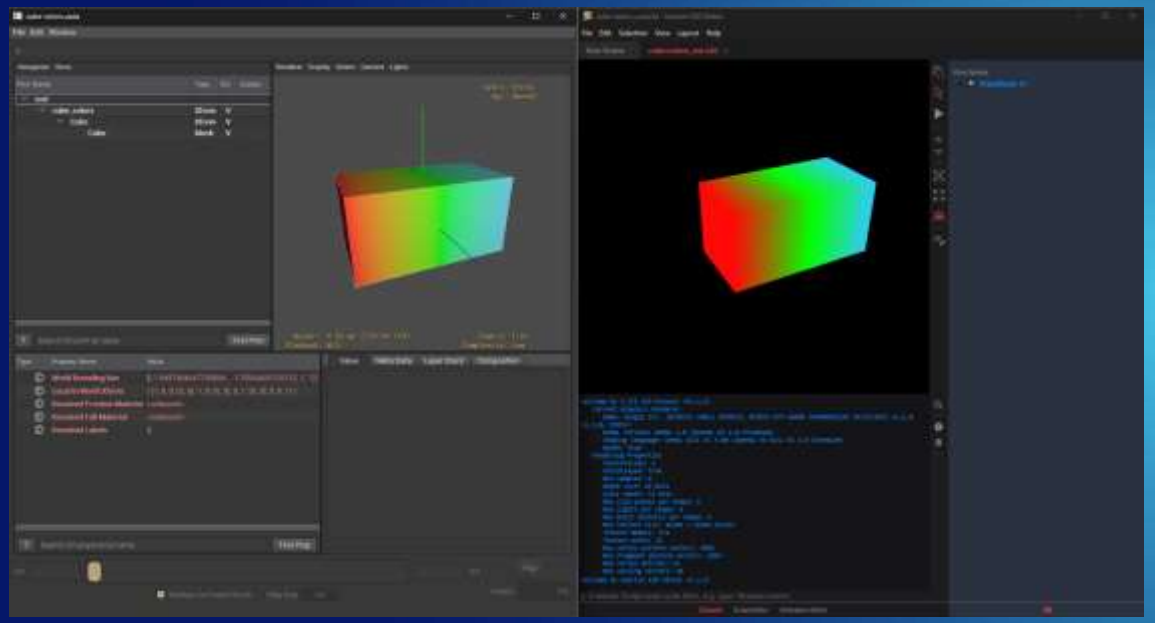
- Foundational layers
- Full field system
- Abstract node interfaces
- Initial tier of concrete nodes
- Working X3D 4.1 XML exporter

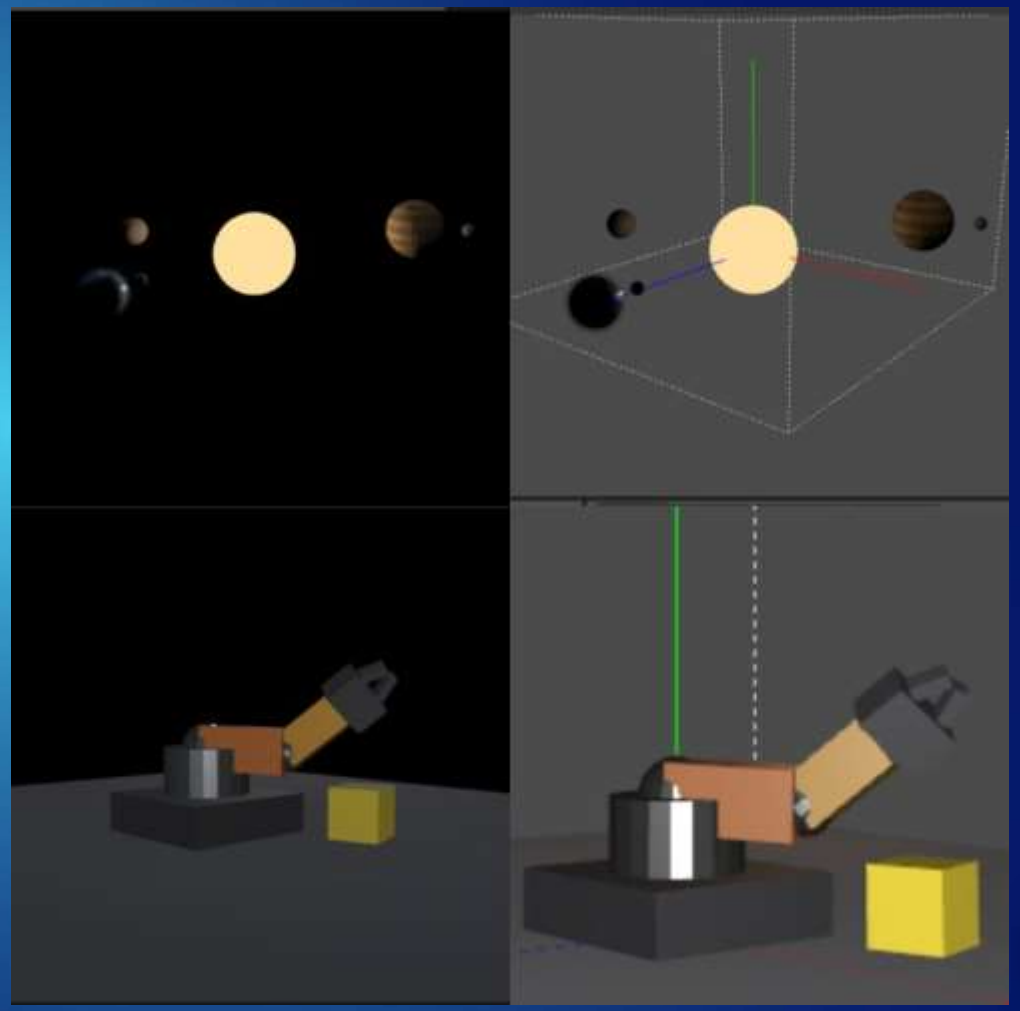
NEXT

- Remaining concrete nodes
- Track 4.1 Committee Draft revisions
- Final binding generated via X3DUOM
- The OpenUSD ⇄ X3D translator



USD-X3D Translator Early Progress





USD-X3D HAnim Translation



X3D



USD



Thank You!

Web3D 2026

Doha, Qatar

