Object Model For X3D (OM4X3D)

Don Brutzman and Roy Walmsley

Summary. The Object Model for X3D (OM4X3D) adds rigor to existing object-oriented interfaces in the X3D Abstract Specification.

This document shows how to trace consistency among X3D references for a given X3D node and field... starting with the Abstract Specification, comparing X3D XML and JavaScript Object Notation (JSON) schemas, all the way through Object Model for X3D (OM4X3D) autogeneration. Then comes autogeneration of X3DJSAIL Java SAI Library and testing with X3DJSONLD Loader. Unit testing with thousands of X3D Examples is confirming correctness. We finish by looking at current work on X3D version 4.

Presentation

Web3D 2017 Conference, Brisbane Australia, 5-7 June 2017
http://web3d2017.web3d.org/program

"Masterclass 4: Object Model for X3D (OM4X3D) and Scene Access Interface (SAI) for X3D, including JSONLD and X3DJSAIL"

References

a. X3D Recommended Standards http://www.web3d.org/standards
   - X3D v3.3 Abstract Specification

b. X3D Specifications: Schema and DOCTYPE Validation http://www.web3d.org/specifications
   - X3D XML Schema Documentation
   - X3D XML DTD Documentation
   - X3D JSON Schema Documentation

c. X3D Resources http://www.web3d.org/x3d/content/examples/X3dResources.html
   - Authoring Support http://www.web3d.org/x3d/content/examples/X3dResources.html#AuthoringSupport

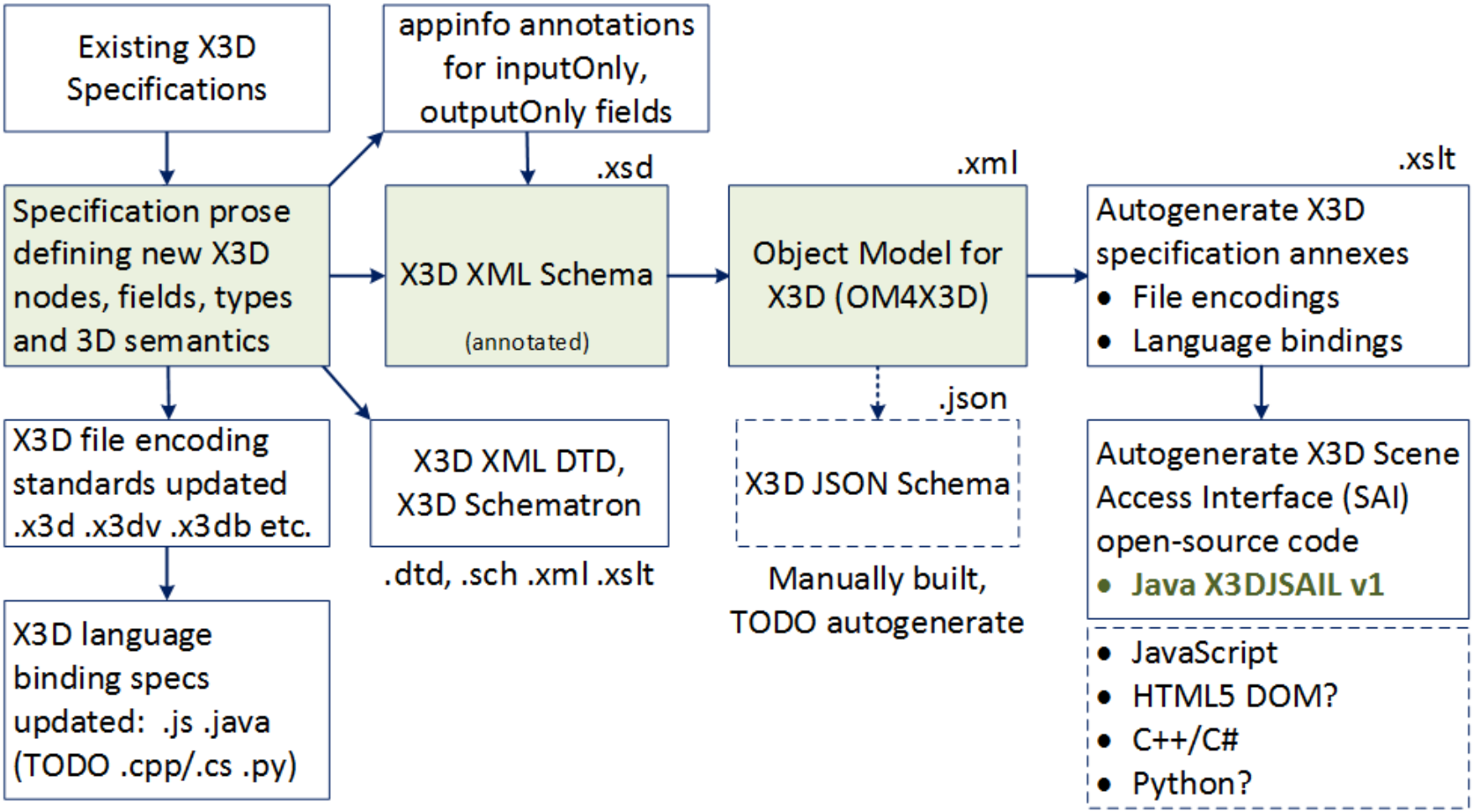
d. X3D Scene Authoring Hints http://www.web3d.org/x3d/content/examples/X3dSceneAuthoringHints.html

e. X3D Tooltips http://www.web3d.org/x3d/content/X3dTooltips.html

f. X3DJSAIL http://www.web3d.org/specifications/java/X3DJSAIL.html

g. X3DJSONLD https://github.com/coderextreme/X3DJSONLD

Object Model for X3D: Creation, Autogeneration



1. ***How to check if some event is missing from the Object Model for X3D (OM4X3D) and related resources?***

On 5/13/2017 9:12 PM, John Carlson wrote:
> @isOver is a field of TouchSensor (but may be in a super class), it is in ObjectModel4X3D, but not in JSON schema 3.3

Response: good observation. I think that is correct. Let's trace it out and cross-check everything.

Continuing to follow this trail illustrates how Object Model for X3D (OM4X3D) gets assembled.

2. ***X3D Abstract Specification***

20.4.4 TouchSensor
http://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/pointingsensor.html#TouchSensor

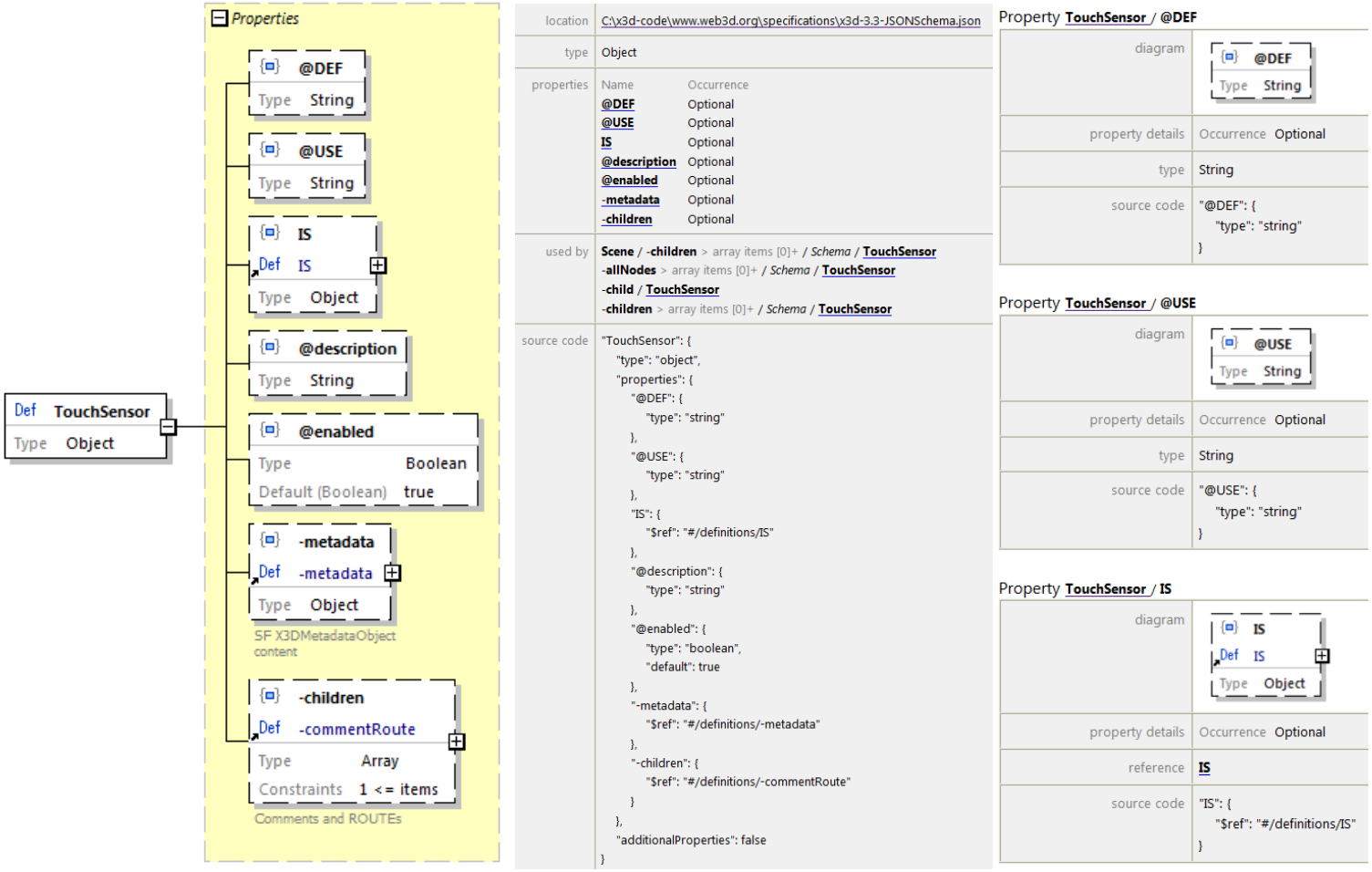
```
================================
TouchSensor : X3DTouchSensorNode {
  SFString [in,out] description         ""
  SFBool   [in,out] enabled             TRUE
  SFNode   [in,out] metadata            NULL [X3DMetadataObject]
  SFVec3f  [out]    hitNormal_changed
  SFVec3f  [out]    hitPoint_changed
  SFVec2f  [out]    hitTexCoord_changed
  SFBool   [out]    isActive
  SFBool   [out]    isOver
  SFTime   [out]    touchTime
}
================================
```

The X3D Abstract Specification is authoritative and confirms accessType, since [out] means outputOnly. Thus isOver is a transient event that can only occur at run time.

3. ***JSON Schema***

JSON Schema: TouchSensor
http://www.web3d.org/specifications/X3dJsonSchemaDocumentation3.3/x3d-3.3-JSONSchema_TouchSensor.html

Since the isOver event has accessType outputOnly, I would not expect a TouchSensor node to include an isOver event in a file encoding. Sure enough, that is the case.



etc.

The preceding link is easily found by going to the X3D Tooltips, which include cross-links to the various types of documentation available.

4. ***X3D Tooltips*** provide summary descriptions and authoring hints for each X3D node (XML element) and field (XML attribute) found in the X3D Specification.

X3D Tooltips: TouchSensor
http://www.web3d.org/x3d/tooltips/X3dTooltips.html#TouchSensor

	accessType and type	Credits and Translations	X3D Resources
TouchSensor	**TouchSensor tracks location and state of the pointing device, detecting when a user points at or selects (activates) geometry.** **Hint: this sensor detects user interactions affecting peer nodes and their child geometry.** **Hint: see X3D Specification 20.2.1 Overview of pointing device sensors** http://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/pointingsensor.html#OverviewOfPointingDeviceSensors **Hint: see X3D Specification 20.2.3 Activating and manipulating pointing device sensors** http://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/pointingsensor.html#Activatingandmanipulating	Search mail lists or Mantis issues	X3D validation: XML Schema, DOCTYPE, JSON Schema

Looking at isOver field which appears under TouchSensor, and then following the _accessType outputOnly_ link:
http://www.web3d.org/x3d/tooltips/X3dTooltips.html#accessType
==========
accessType Definitions

accessType determines whether a field corresponds to event input, event output, or persistent state information. Events are strictly typed values with a corresponding timestamp. ROUTE connections must match accessType between source field and target field.

* *initializeOnly:* can be initialized, but cannot send or receive events. This is usually the case for fields that are considered too computationally expensive to change at run time.
* *inputOutput:* can be initialized, and can also send or receive events.
* *inputOnly:* cannot be initialized or included in a scene file, but can receive input event values via a ROUTE.
* *outputOnly:* cannot be initialized or included in a scene file, but can send output event values via a ROUTE.

X3D accessType design keeps 3D graphics rendering fast and interactive, also helping to keep X3D players small and lightweight.
==========

5. ***X3D validation using XML DTD and XML Schema.*** When we look at X3D DTD and X3D XML Schema documentation, similar file-validation constructs are found.

DOCTYPE: X3D DTD

Generated DTD Grammar Documentation for: x3d-3.3 - TouchSensor
http://www.web3d.org/specifications/X3dDoctypeDocumentation3.3.html#TouchSensor

TouchSensor

Declared Attributes

- #IMPLIED CDATA description
- #DEFAULT ENUMERATION (true | false) enabled = true
- #DEFAULT NMTOKEN containerField = children
- #IMPLIED NMTOKENS class
- #IMPLIED ID DEF
- #IMPLIED IDREF USE

Element Content Model

((IS?), (MetadataBoolean | MetadataDouble | MetadataFloat | MetadataInteger | MetadataSet | MetadataString | ProtoInstance)?)

Referenced by

Anchor, Billboard, Collision, field, fieldValue, Group, LOD, ProtoBody, Scene, StaticGroup, Switch, Transform, EspduTransform, GeoLocation, GeoLOD, GeoTransform, HAnimHumanoid, HAnimSegment, HAnimSite, CADAssembly, CADLayer, Layer, Viewport, LayoutGroup, LayoutLayer, ScreenGroup, PickableGroup

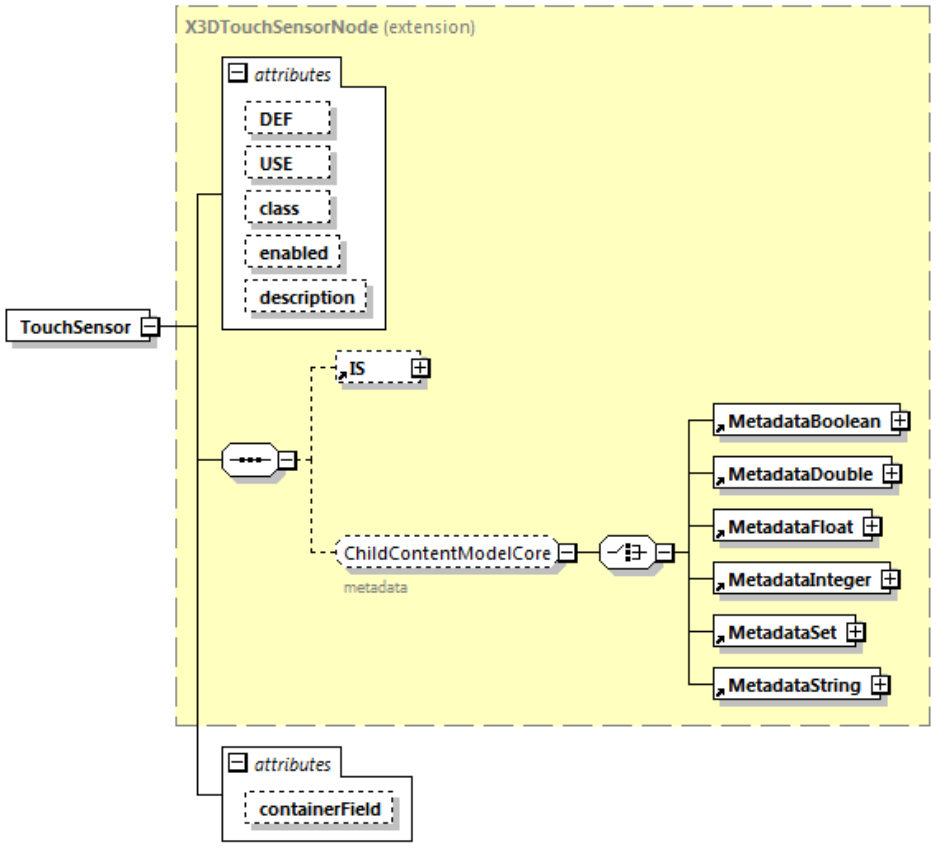
Note that isOver is not present here. Thus if it appears in file content, something like <*TouchSensor isOver='false'*/> will fail validation.

Similarly for **X3D XML Schema validation**

X3D XML Schema x3d-3.3.xsd documentation: TouchSensor
http://www.web3d.org/specifications/X3dSchemaDocumentation3.3/x3d-3.3_TouchSensor.html

once again shows description and enabled fields, but no isOver field.

X3D XML Schema



type	extension of **X3DTouchSensorNode**					
properties	content complex					
children	**IS MetadataBoolean MetadataDouble MetadataFloat MetadataInteger MetadataSet MetadataString**					
used by	group **ChildContentModelInteractive**					
attributes	Name	Type	Use	Default	Fixed	Annotation
	DEF	**xs:ID**				appinfo DEF defines a unique ID name for each node, referencable by other nodes.
	USE	**xs:IDREF**				appinfo USE means reuse an already DEF-ed node ID, excluding all child nodes and all other attributes (except for containerField, which can have a different value).
	class	**xs:NMTOKENS**				appinfo The class attribute is a space-separated list of classes, reserved for use by XML stylesheets.
	enabled	**SFBool**		true		
	description	**SFString**				
	containerField	**xs:NMTOKEN**		children		

6. ***X3D XML Schema annotations.*** So how does isOver find its way into OM4X3D?

Answer: information regarding inputOnly/outputOnly fields is put into each node's annotation/appinfo section in the X3D XML Schema. There it is similar to a comment - actually structured metadata - that can be read and converted when creating OM4X3D. Continuing the journey:

X3D XML Schema file
http://www.web3d.org/specifications/x3d-3.3.xsd

```
==========================================
<xs:element name="TouchSensor">
    <xs:annotation>
        <xs:appinfo>
            <xs:attribute name="hitNormal_changed" type="SFVec3f" fixed="outputOnlyField"/>
            <xs:attribute name="hitPoint_changed" type="SFVec3f" fixed="outputOnlyField"/>
            <xs:attribute name="hitTexCoord_changed" type="SFVec2f" fixed="outputOnlyField"/>
            <xs:attribute name="componentName" type="xs:NMTOKEN"
fixed="PointingDeviceSensor"/>
            <xs:attribute name="componentLevel" type="xs:positiveInteger" fixed="1"/>
        </xs:appinfo>
        <xs:documentation source="http://www.web3d.org/documents/specifications/19775-
1/V3.3/Part01/components/pointingsensor.html#TouchSensor"/>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="X3DTouchSensorNode">
                <xs:attribute name="containerField" type="xs:NMTOKEN" default="children"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
==========================================
```

Hmm, now what? Well, in addition to decorating each node in the X3D XML Schema with inputOnly/outputOnly fields, we have to deal with inheritance. As previously noted, that specifically means handling superclass information.

7. ***Object oriented relationships.*** Object-oriented (O-O) constructs appear throughout X3D Abstract Specification, X3D XML Schema, and XML Schema guidance.

Inspection of the preceding XML shows how the XML Schema takes advantage of object-oriented inheritance, matching the node types found in the X3D Abstract Specification.

For example: as we saw far above, TouchSensor implements the X3DTouchSensorNode interface. It still looks like the following:

20.3.3 X3DTouchSensorNode
http://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/pointingsensor.html#X3DTouchSensorNode

```
========================================
X3DTouchSensorNode : X3DPointingDeviceSensorNode {
  SFString [in,out] description ""
  SFBool   [in,out] enabled     TRUE
  SFNode   [in,out] metadata    NULL [X3DMetadataObject]
  SFBool   [out]    isActive
  SFBool   [out]    isOver
  SFTime   [out]    touchTime
}
========================================
```

The X3D Abstract Specification section 4.4.2.3 Interface hierarchy includes "ascii text art" for the interface hierarchy. It is the authoritative reference for the object model.

> *"Most object types derive some of their interfaces and functionality from other object types in the system. These are known as its supertypes, and an object is said to be derived from these supertypes. Likewise, these supertypes may derive their capabilities from other object types, forming a chain all the way to a small number of base types from which all the others are ultimately derived. The graph describing the relationship between all object types in the system is called the interface hierarchy. In this part of ISO/IEC 19775, the object hierarchy specifies conceptual relationships between objects but does not necessarily dictate actual implementation."*

excerpt: [X3D Abstract Specification 4.4.2.3 Interface hierarchy]

```
X3DNode
  |
  +- Contact
  +- Contour2D
  +- EaseInEaseOut
  +- GeoOrigin (deprecated)
  +- LayerSet
  +- MetadataBoolean (X3DMetadataObject)*
  +- MetadataDouble (X3DMetadataObject)*
  +- MetadataFloat (X3DMetadataObject)*
  +- MetadataInteger (X3DMetadataObject)*
  +- MetadataSet (X3DMetadataObject)*
  +- MetadataString (X3DMetadataObject)*
  +- NurbsTextureCoordinate
  +- RigidBody
  +- ShaderPart (X3DUrlObject)*
  +- ShaderProgram (X3DUrlObject, X3DProgrammableShaderObject)*
  +- TextureProperties
  |
  +- X3DAppearanceNode -+- Appearance
  |
  +- X3DAppearanceChildNode -+- FillProperties
  |                          +- LineProperties
  |                          |
  |                          +- X3DMaterialNode -+- Material
  |                          |                   +- TwoSidedMaterial
  |                          |
  |                          +- X3DShaderNode -+- ComposedShader (X3DProgrammableShaderObject)*
  |                          |                 +- PackagedShader (X3DUrlObject, X3DProgrammableShaderObject)*
  |                          |                 +- ProgramShader
  |                          |
  |                          +- X3DTextureNode -+- MultiTexture
  |                          |                  |
  |                          |                  +- X3DEnvironmentTextureNode -+- ComposedCubeMapTexture
  |                          |                  |                             +- GeneratedCubeMapTexture
  |                          |                  |                             +- ImageCubeMapTexture (X3DUrlObject)*
  |                          |                  |
  |                          |                  +- X3DTexture2DNode -+- ImageTexture (X3DUrlObject)*
  |                          |                  |                    +- MovieTexture (X3DSoundSourceNode, X3DUrlObject)*
  |                          |                  |                    +- PixelTexture
  |                          |                  |
  |                          |                  +- X3DTexture3DNode -+- ComposedTexture3D
  |                          |                                       +- ImageTexture3D (X3DUrlObject)*
  |                          |                                       +- PixelTexture3D
  |                          |
  |                          +- X3DTextureTransformNode  +- MultiTextureTransform
  |                                                     -+- TextureTransform
  |                                                      +- TextureTransformMatrix3D
  |                                                      +- TextureTransform3D
```

A further excerpt:

```
  +- X3DSensorNode -+- Collision (X3DGroupingNode)*
  |                 +- CollisionSensor
  |                 |  [intermediate values omitted]
  |                 |
  |                 +- X3DPointingDeviceSensorNode -+- X3DDragSensorNode -+- CylinderSensor
  |                                                 |                      +- PlaneSensor
  |                                                 |                      +- SphereSensor
  |                                                 |
  |                                                 +- X3DTouchSensorNode -+- GeoTouchSensor
                                                                           +- TouchSensor
```

So TouchSensor indeed has a hierarchy of parent interfaces above it…

As part of representing these X3D Abstract Specification relationships, the XML Schema includes this abstract node type:

```
<xs:complexType name="X3DTouchSensorNode" abstract="true">
    <xs:annotation>
        <xs:appinfo>
            <xs:attribute name="touchTime" type="SFTime" fixed="outputOnlyField"/>
            <xs:attribute name="componentName" type="xs:NMTOKEN"
fixed="PointingDeviceSensor"/>
            <xs:attribute name="componentLevel" type="xs:positiveInteger" fixed="1"/>
            Base type for all touch-style pointing device sensors.
        </xs:appinfo>
        <xs:documentation source="http://www.web3d.org/documents/specifications/19775-
1/V3.3/Part01/components/pointingsensor.html#X3DTouchSensorNode"/>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="X3DPointingDeviceSensorNode"/>
    </xs:complexContent>
</xs:complexType>
```

All well and good. But object-oriented relationships can be difficult to work with if you want to process that information yourself. There are a lot of sophisticated constructs in XML Schema that we usually don't have to worry about.

Multiple authoritative references for XML Schema validation of XML content can be found at:

W3C XML Schema
https://www.w3.org/XML/Schema

Assessment: XML Schema works great for XML validation of scenes. It also works well for formal codification of additional non-XML information in the X3D Abstract Specification, such as X3D accessType values for each field. However the object-oriented inheritance within that schema .xsd XML file it is pretty difficult to parse and process thoroughly, if you want to take advantage of the information for other purposes.

So we keep going... How to provide all of this essential information in a more useful form?

8. ***Object Model for X3D (OM4X3D) construction and result. ***

We want a single XML file that codifies all the X3D Abstract Specification information, but lists all information for each node in one place for easier processing.

So, we have produced an XSLT stylesheet that reads the X3D XML Schema (itself in XML) to create the OM4X3D file.

BuildObjectModelXmlFile.xslt
http://www.web3d.org/x3d/stylesheets/BuildObjectModelXmlFile.xslt
https://sourceforge.net/p/x3d/code/HEAD/tree/www.web3d.org/x3d/stylesheets/BuildObjectModelXmlFile.xslt

As shown, and as with all other resources here, each asset is placed in version control for explicit/easy/shared editing.

The resulting OM4X3D file can be found at

X3DObjectModel-3.3.xml
http://www.web3d.org/specifications/X3DObjectModel-3.3.xml
https://sourceforge.net/p/x3d/code/HEAD/tree/www.web3d.org/specifications/X3DObjectModel-3.3.xml

So here is the OM4X3D excerpt for the TouchSensor node entry, in all its complete and verbose glory:

```
<ConcreteNode name="TouchSensor">
   <InterfaceDefinition specificationUrl=
      "http://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/pointingsensor.html#TouchSensor">

       <componentInfo name="PointingDeviceSensor" level="1"/>
       <Inheritance baseType="X3DTouchSensorNode"/>
       <field type="SFString"
              accessType="inputOutput"
              name="description"
              inheritedFrom="X3DPointingDeviceSensorNode"/>
       <field type="SFBool"
              accessType="inputOutput"
              name="enabled"
              default="true"
              inheritedFrom="X3DSensorNode"/>
       <field type="SFVec3f" accessType="outputOnly" name="hitNormal_changed"/>
       <field type="SFVec3f" accessType="outputOnly" name="hitPoint_changed"/>
       <field type="SFVec2f" accessType="outputOnly" name="hitTexCoord_changed"/>
       <field type="SFBool"
              accessType="outputOnly"
              name="isActive"
              inheritedFrom="X3DSensorNode"/>
       <field type="SFBool"
              accessType="outputOnly"
              name="isOver"
              inheritedFrom="X3DPointingDeviceSensorNode"/>
```

```
      <field type="SFNode"
             accessType="inputOutput"
             name="metadata"
             default="NULL"
             acceptableNodeTypes="X3DMetadataObject"
             inheritedFrom="X3DNode"/>
      <field type="SFTime"
             accessType="outputOnly"
             name="touchTime"
             inheritedFrom="X3DTouchSensorNode"/>
      <field type="SFString"
             accessType="inputOutput"
             name="DEF"
             inheritedFrom="DEF_USE"/>
      <field type="SFString"
             accessType="inputOutput"
             name="USE"
             inheritedFrom="DEF_USE"/>
      <field type="SFString"
             accessType="inputOutput"
             name="class"
             inheritedFrom="globalAttributes"/>
      <containerFieldDefault name="children"/>
      <ContentModel>
         <GroupContentModel name="ChildContentModelCore" minOccurs="0"/>
      </ContentModel>
   </InterfaceDefinition>
</ConcreteNode>
```

Please do check closely. Note that all field information (attributes and child nodes) appears to be present in support of TouchSensor.

This kind of fully detailed information is provided for each and every node in the X3D Specification. Exhaustive declaration of all attribute and element fields for each node actually makes adaptive use of this information much easier for custom processors to handle.

9. ***Simple Example Test.*** A simple example conversion test actually creates a useful product: All X3D Elements Attributes inventories.

- AllX3dElementsAttributes3.3.txt and
- AllX3dElementsAttributes3.3.xml (if using Web browser, view source)

These two files are produced by reading the OM4X3D file and processing it with the following exemplar XSLT stylesheet:

- AllX3dElementsAttributesTextTemplate.xslt

These autogenerated inventory files are intended to be repeatable. Software developers can edit/convert these files in any manner they like, or even adapt the stylesheet. This simple approach is quite useful for creating new X3D-aware software, documentation, and other products. Note that default attribute values (which are optional in an .x3d scene) are also found in the included information.

Excerpt:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- All X3D elements and attributes, with default values, as defined in X3D Schema version 3.3 -->

<AllX3dElementsAttributes>

    <Anchor bboxCenter='0 0 0' bboxSize='-1 -1 -1' description='' parameter='' url='' containerField='children'/>

    <Appearance containerField='appearance'/>

    <Arc2D endAngle='1.570796' radius='1' startAngle='0' containerField='geometry'/>

    <ArcClose2D closureType='PIE' endAngle='1.570796' radius='1' solid='false' startAngle='0' containerField='geometry'/>

    <AudioClip description='' loop='false' pauseTime='0' pitch='1.0' resumeTime='0' startTime='0' stopTime='0' url=''
    containerField='source'/>

    <Background backUrl='' bottomUrl='' frontUrl='' groundAngle='' groundColor='' leftUrl='' rightUrl='' skyAngle=''
    skyColor='0 0 0' topUrl='' transparency='0' containerField='children'/>

    <BallJoint anchorPoint='0 0 0' forceOutput='"NONE"' containerField='joints'/>

    <Billboard axisOfRotation='0 1 0' bboxCenter='0 0 0' bboxSize='-1 -1 -1' containerField='children'/>

    [… etc…]
```

10. ***Applying OM4X3D to build X3DJSAIL.*** Well OK, quite a pile of information, but is it truly correct and does it really work?

Verifying that all of the information here is indeed correct and complete is beyond the ability of us mere mortals. However tools can help.

The OM4X3D is used, in turn, to autogenerate a complete API for X3D. Subsequent compilation and unit-test verification of thousands of X3D example scenes really puts it through the wringer to confirm correctness and completeness.

X3D Java Scene Access Interface Library (X3DJSAIL)
http://www.web3d.org/specifications/java/X3DJSAIL.html

How did TouchSensor come out? Two ways: first original X3D SAI interfaces and then compatible concrete classes are generated.

Abstract Interfaces:
* org.web3d.x3d.sai.PointingDeviceSensor
http://www.web3d.org/specifications/java/javadoc/org/web3d/x3d/sai/PointingDeviceSensor/X3DTouchSensorNode.html
* org.web3d.x3d.sai.PointingDeviceSensor.TouchSensor
http://www.web3d.org/specifications/java/javadoc/org/web3d/x3d/sai/PointingDeviceSensor/TouchSensor.html

Concrete (instantiable) classes:
* org.web3d.x3d.jsail.PointingDeviceSensor.TouchSensorObject
http://www.web3d.org/specifications/java/javadoc/org/web3d/x3d/jsail/PointingDeviceSensor/TouchSensorObject.html

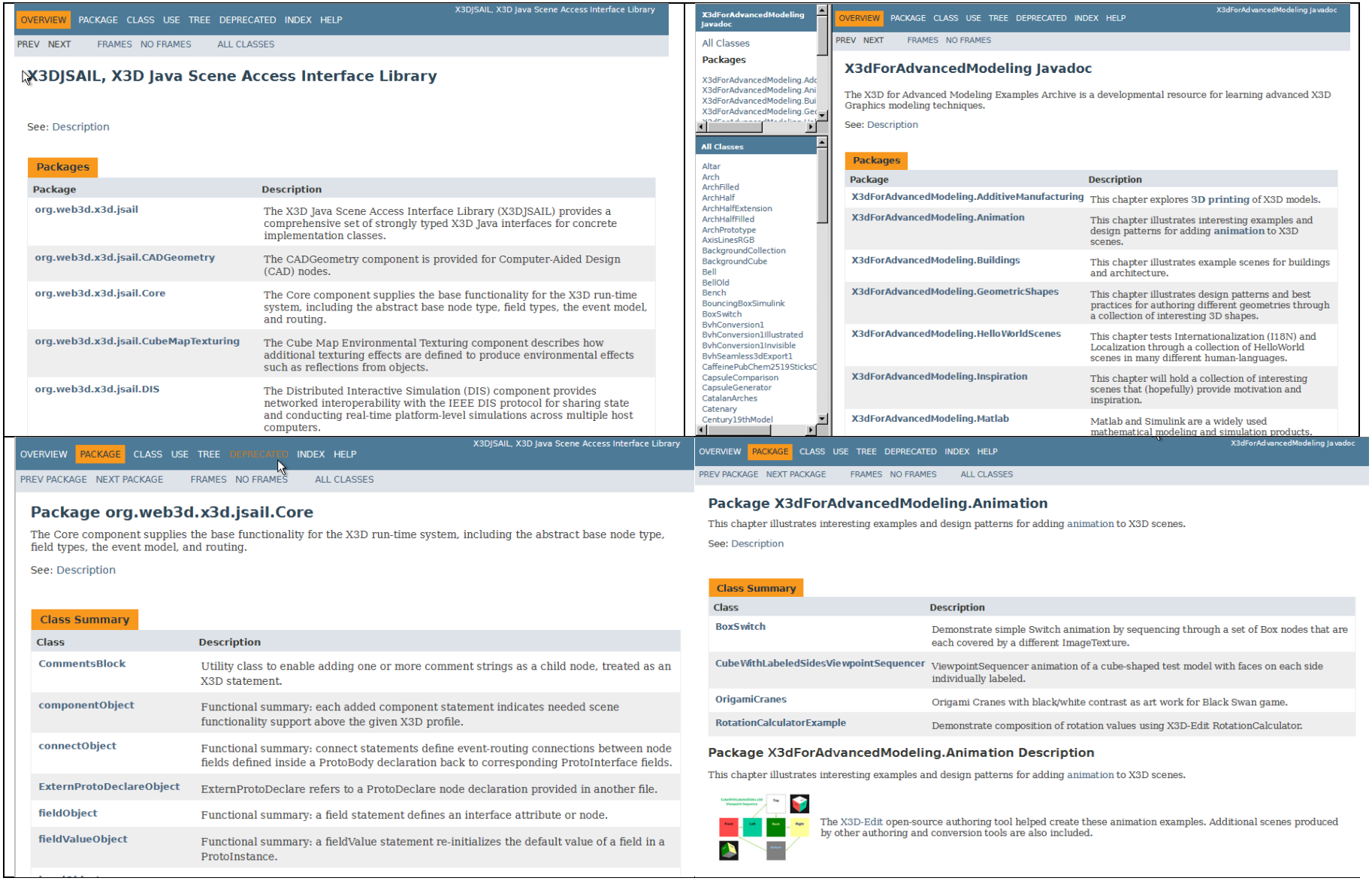
* TouchSensorObject isOver
http://www.web3d.org/specifications/java/javadoc/org/web3d/x3d/jsail/PointingDeviceSensor/TouchSensorObject.html#method.summary
http://www.web3d.org/specifications/java/javadoc/org/web3d/x3d/jsail/PointingDeviceSensor/TouchSensorObject.html#getIsOver--
"Provide boolean value from outputOnly SFBool field named isOver."

and then, following several XSLT stylesheet loop-de-loops, the original X3D Schema descriptive annotations and corresponding X3D Tooltip are each extracted and inserted into the Javadoc documentation. Excerpt:

"Tooltip: Hover over geometry by aiming the mouse (or pointing device) to generate isOver events. Sensor sends output event isOver=true event when pointing device moves over sensor's geometry, and later sends output event isOver=false event when pointing device moves off."

So it looks like a huge round trip is completed and cross-verifying itself. Win win win win win win. No really! ☺

Javadoc tells a really big story when autogenerating the X3DJSAIL library as well as 3900 open-source scenes in X3D Examples Archives.



This open build process provides amazing (and repeatable) breadth and depth for unit testing for [X3D Quality Assurance (QA)](#).

11. ***OM4X3D milestones.*** Lots of excellent milestones are now occurring, further enabling X3D v4 progress.

Thousands of example scenes in the X3D Examples Archives provide an amazing set of unit tests. Consistent validation and outputs and round-trip testing is being performed for

- Multiple encodings: XML, ClassicVRML, VRML97, JavaScript Object Notation (JSON), Compressed Binary Encoding (CBE).
- Language bindings: Java compilation and execution self-test validation. C++ (and possibly Python) is expected in the future.
- http://www.web3d.org/x3d/content/examples/X3dResources.html#Examples (current inventory: 3897 X3D scenes).
- Display and interaction testing using Cobweb and X3DOM players is provided directly.
- Additional testing regularly occurs in multiple other X3D applications and players.
- X3D Quality Assurance (QA) and X3D Validator helps test that scenes are correct, avoiding false errors from erroneous content.

We are now checking edge conditions and corner cases, as well as correspondingly autogenerated JSON-encoded scenes, to verify that indeed all of this is hanging together satisfactorily. **No showstoppers found.**

We are optimistic that the specification review process can collaboratively agree on the following:

a. OM4X3D and X3D XML Schema correctly capture X3D Abstract Specification relationships.
b. X3D Java SAI implementation by X3DJSAIL beta testing is satisfactorily complete, and codebase is ready for initial release v1.0.
c, X3D JSON encoding design is complete and correct, again as verified by extensive unit testing with X3D Example Archives.

Looking ahead. Given that all of the version-control autogeneration capabilities described here currently (or easily can) work for all versions of X3D versions 3.0 through 3.3, it is a straightforward task to extend all new nodes and every single asset to X3D version 4.

Strict formality plus interface autogeneration will greatly accelerate our ability to produce a rigorously grounded X3D v4 specification.

X3D Version 4
http://www.web3d.org/x3d4
X3D Version 4 Development
http://www.web3d.org/wiki/index.php/X3D_version_4.0_Development

Thanks for many many efforts by many many contributors, all coming together in grand style.

All feedback and improvements welcome. Thank you for considering the possibilities. Have fun with OM4X3D! 8)