

# X3D Graphics for Web Authors

## Chapter 14

### Creating Prototype Nodes

*There are more things in heaven and earth, Horatio,  
than are dreamt of in your philosophy.*

William Shakespeare, Hamlet Act I Scene V

# Contents

Chapter Overview and Concepts

Functional Descriptions and Examples

Chapter Summary

Suggested Exercises

References

# Chapter Overview

# Overview: Prototypes

## Concepts

- Motivation and Functional Summary

## Functional Descriptions and Examples

- ProtoDeclare, ProtoInterface, ProtoBody and field declarations
- IS / connect linking of field interfaces to internals
- ExternProtoDeclare and field signatures
- ProtoInstance, containerField, fieldValue initializations
- Advanced examples: design and re-use

[back to Table of Contents](#)

# Concepts

# Prototype motivation: extensibility

The X in X3D stands for Extensible: we have engineered the X3D standard for future growth

- Supporting innovation by individual authors, rather than waiting for future versions of the specification

Other extensibility mechanisms available:

- Inline node allows one scene to pull in other scenes, but without modification or customization
- Script node allows creation of arbitrary functionality that receives (and responds to) routed events

Prototypes create new full-fledged X3D nodes

- With field definitions, render capability, etc.

# Comparison with Inline node

Inline is easier to create and use

- Simply loads and inserts another X3D scene

Inline nodes are less flexible

- Cannot be customized when imported since there is no override mechanism for internal field values
- Events can be passed into, out of Inline scene at run time by using predefined IMPORT, EXPORT statements, for exposed internal nodes inside Inline

Prototypes are preferred if initialization values are needed, routing also works unambiguously

# Prototype functional summary

A Prototype creates a new full-fledged X3D node

- With field definitions, render capability, etc.

X3D prototypes provide a way for X3D authors to create new node definitions

- ProtoInstance allows repeated reuse of a new node
- Fields can be exposed and parameterized, allowing customization (unlike Inline which is fixed content)

Prototypes can be used within the scene where they are defined, or used externally

- ExternProtoDeclare gives reference to declaration

# Declaration versus instances

Prototype declarations can be thought of as defining a cookie-cutter for a new node

- ProtoDeclare constructs the definition
- Definition does not yet create an actual new node

Prototype instances are the actual copies of the new node which gets displayed

- Just as cookie cutter is used to create new cookies

ProtoDeclare  
is a  
template



ProtoInstance  
copies actually  
exist and render

# Summary of xml element structure

## ProtoDeclare

- ProtoInterface
  - field
- ProtoBody
  - Initial node
  - Additional nodes
  - IS/connect links

## Defines prototype

- Hold field definitions
  - Defines each field interface
- Hold nodes, scene subgraph
  - First node defines type, use
  - Initial siblings not rendered
  - Link interfaces to internal fields

## ExternProtoDeclare

- field

## Retrieve external declaration

- List of fields without values

## ProtoInstance

- fieldValue

## Actual copy of prototype node

- Override default interface values

# Potential power

Prototypes are a powerful technique for extending the capabilities of X3D

Few computing languages provide authors with the capability to extend the core vocabulary of the language itself

In one sense, an scene author defining a prototype for a new node in a scene can be thought to have similar power as the X3D specification team which defines new nodes for everyone to use in X3D

# Strong typing of nodes

Each prototype declaration must contain at least one node in the prototype body

- First node is primary, defining type for prototype
- ProtoInstances can only appear where that primary node might be allowed to appear
- If primary node contains children, together they must define a valid scene subgraph

Subsequent sibling nodes can follow first node

- But are not rendered, nor do they affect node type

Thus prototype instances remain strongly typed

- Any errors are discoverable before run time

# Syntax alert: contrast .x3d .x3dv

Syntax for prototype definition and usage is significantly different when comparing the XML (.x3d) and ClassicVRML (.x3dv) encodings

Functional correspondence remains identical

- Declaration, field definitions, instance creation, etc.

Book compares both forms of syntax in detail

# Functional Descriptions and Examples

# ProtoDeclare

A prototype declaration includes two constructs:  
prototype interface and prototype body

```
<ProtoDeclare name='MyNewBlueMaterial'>
```

```
<ProtoInterface>
```

```
  <field name='concentration' accessType='inputOutput' type='SFInt32'  
    Value='0.75' appinfo='how blue is my new Material, range 0..1'>
```

```
</ProtoInterface>
```

```
<ProtoBody>
```

```
  <!-- First node in body determines node type of prototype-->
```

```
  <Material/>
```

```
  <!-- Subsequent nodes do not render, but must be valid X3D subgraph -->
```

```
  <Script DEF='CalculateNewBlueValueFromConcentration'>
```

```
</ProtoBody>
```

```
</ProtoDeclare>
```

# Naming considerations 1

Good naming is important for prototypes, fields

- Helps authors understand their intent and then utilize them correctly
- Naming-convention guidelines found in X3D Scene Authoring Hints

Only one declaration is allowed for each individual prototype node

- Cannot have conflicting same-name definitions from ProtoDeclare and/or ExternProtoDeclare
- Name collisions (i.e. “overloading”) not allowed

# Naming considerations 2

Good test of a prototype name (or field name) is to use it in a sentence, to see if it makes sense

- “a MaterialModulator node mimics a Material node and modulate fields as an animation effect”
- Awkward names are revealed by awkward sentences
- Descriptions are helpful when added as *appinfo*

Good names provide clarity when thinking about, modifying, and debugging a scene

Best name is when no one asks what it means!

- Alternatively, questions imply need to improve

# Naming conventions, excerpted

CamelCaseNaming: capitalize each word, never use abbreviations, strive for clarity, be brief but complete  
startWithLowerCaseLetter when defining field names (i.e. attributes) for Prototypes, Scripts

Ensure consistent capitalization throughout

Use the underscore character ("\_") to indicate subscripts on mathematical variables. Otherwise avoid use of underscores since they look like whitespace when part of a URL address

Avoid use of hyphens ("-") since these are erroneously turned into subtraction operators when converted into class or variable names

# ProtoInterface and field declarations

<ProtoInterface> is section of <ProtoDeclare> that holds <field> definitions

- Which are the interface for the prototype
- Zero or more <field> definitions allowed
- <ProtoInterface> omitted if no <field> definitions

Same as <field> definitions for Script node

- Defines *name*, *type*, *accessType*, and initial *value*
- SFNode, MFNode initializations are contained elements
- initializeOnly, inputOutput fields must have initial value
- inputOnly, outputOnly fields have no initial value

Field-Type Names	Description	Default Values
SFBool	Single-Field boolean value	false (XML syntax) or FALSE (ClassicVRML syntax)
MFBool	Multiple-Field boolean array	Empty list
SFColor	Single-Field color value, RGB	0 0 0
MFColor	Multiple-Field color array, RGB	Empty list
SFColorRGBA	Single-Field color value, red-green-blue alpha (opacity)	0 0 0 0
MFColorRGBA	Multiple-Field color array, red-green-blue alpha (opacity)	Empty list
SFInt32	Single-Field 32-bit integer value	0
MFInt32	Multiple-Field 32-bit integer array	Empty list
SFFloat	Single-Field single-precision floating-point value	0.0
MFFloat	Multiple-Field single-precision floating-point array	Empty list
SFDouble	Single-Field double-precision floating-point value	0.0
MFDouble	Multiple-Field double-precision array	Empty list
SFImage	Single-Field image value	0 0 0 Contains special pixel-encoding values, see Chapter 5 for details

MFIImage	Multiple-Field image value	Empty list
SFNode	Single-Field node	Empty node, NULL
MFNode	Multiple-Field node array of peers	Empty list
SFRotation	Single-Field rotation value using 3-tuple axis, radian-angle form	0 0 1 0
MFRotation	Multiple-Field rotation array	Empty list
SFString	Single-Field string value	Empty string, representable as two adjacent quotation marks
MFString	Multiple-Field string array	Empty list
SFTime	Single-Field time value	-1, sentinel indicating no time value.
MFTime	Multiple-Field time array	Empty list
SFVec2f/SFVec2d	Single-Field 2-float/2-double vector value	0 0
MFVec2f/MFVec2d	Multiple-Field 2-float/2-double vector array	Empty list
SFVec3f/SFVec3d	Single-Field vector value of 3-float/3-double values	0 0 0
MFVec3f/MFVec3d	Multiple-Field vector array of 3-float/3-double values	Empty list

# ProtoBody

First node in ProtoBody is required and critical, defining the node type

- This node is how a ProtoInstance will appear to scene graph

Additional nodes are allowed, but not rendered

- This is how prototypes provide extensibility while maintaining strong node typing
- X3D-Edit will provide warning about this, unless author inserts a comment beforehand

No object-oriented “inheritance” but...

- first node in body can be a nested ProtoInstance

# Simple example: UniversalMedia excerpt 1

The Universal Media Materials archive provides a number of example materials

- Available as prototypes, or cut + paste
- Built in, selectable within X3D-Edit Material editor
- No ProtoInterface/fields needed, just ProtoBody

```
<ProtoDeclare name='ArtDeco00'>
  <ProtoBody>
    <Material ambientIntensity='0.25'
      diffuseColor='0.282435 0.085159 0.134462'
      emissiveColor='0.0 0.0 0.0' shininess='0.127273'
      specularColor='0.276305 0.11431 0.139857' transparency='0.0'!>
    </ProtoBody>
  </ProtoDeclare>
```

# Simple example: UniversalMedia excerpt 2

## Alternatively, ExternProto retrieval:

```
<ExternProtoDeclare name='ArtDeco00'  
    url=""ArtDecoPrototypesExcerpt.x3d#ArtDeco00"  
"http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-  
Prototypes/ArtDecoPrototypesExcerpt.x3d#ArtDeco00"  
"http://www.web3d.org/x3d/content/examples/Basic/UniversalMediaMaterials/  
ArtDecoPrototypes.x3d#ArtDeco00"/>
```

## Invocation is identical in either case:

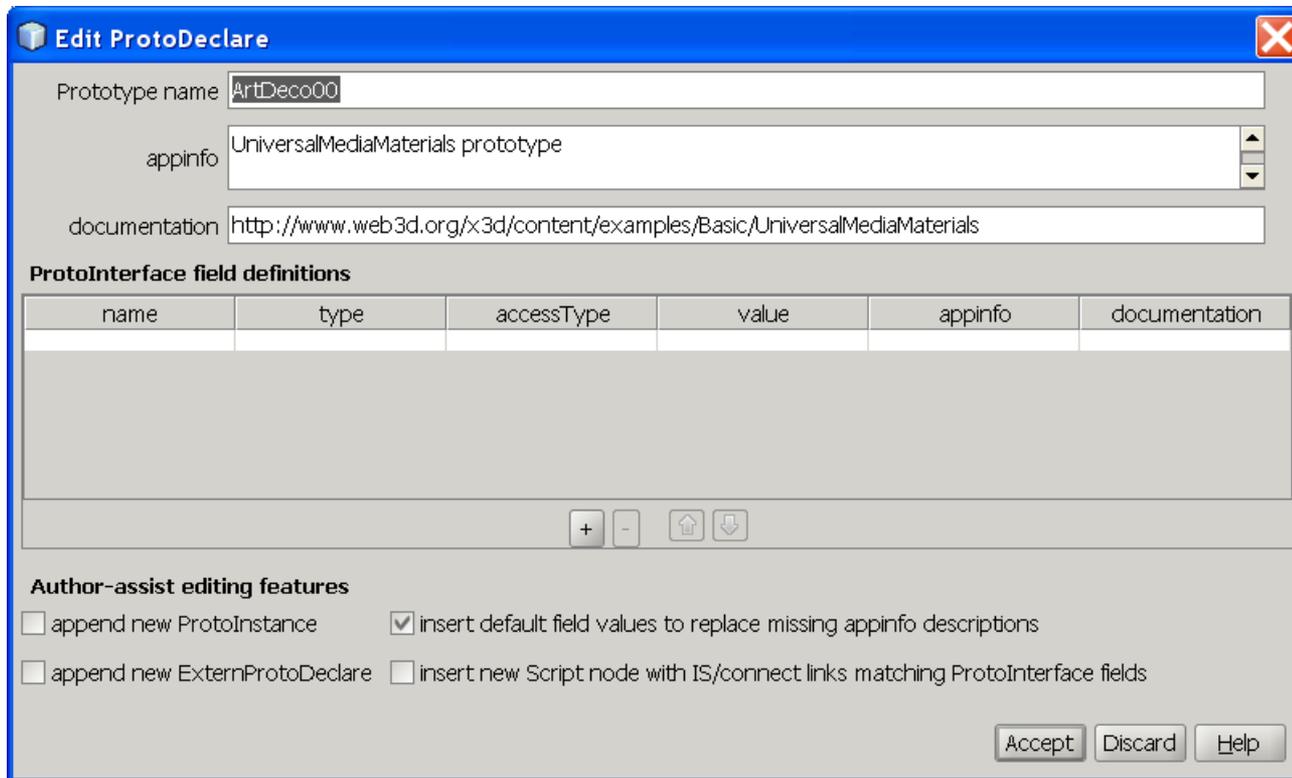
```
<Shape>  
  <Appearance>  
    <ProtoInstance containerField='material' name='ArtDeco00'/>  
  </Appearance>  
  <Sphere DEF='Ball' radius='0.5'/>  
</Shape>
```

*containerField* tells parent node the node type of the contained ProtoInstance.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
3 <X3D profile='Immersive' version='3.0' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/sp
4 <head>
5   <meta content='ArtDecoPrototypesExcerpt.x3d' name='title'/>
6   <meta content='Prototype declarations defining values for X3D/VRML materials, originally converted from SGI&apos;s Open Inventor material exampl
7   <meta content='David Roussel' name='creator'/>
8   <meta content='James Harney, Don Brutzman NPS' name='translator'/>
9   <meta content='7 April 2002' name='created'/>
10  <meta content='18 November 2008' name='modified'/>
11  <meta content='http://vrmilstuff.free.fr/materials' name='reference'/>
12  <meta content='Universal Media Material Library' name='subject'/>
13  <meta content='http://www.web3d.org/x3d/content/examples/Basic/UniversalMediaMaterials/ArtDecoPrototypes.x3d' name='reference'/>
14  <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ArtDecoPrototypesExcerpt.x3d' name='identifier'/>
15  <meta content='Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html' name='generator'/>
16  <meta content='../license.html' name='license'/>
17 </head>
18 <Scene>
19   <ProtoDeclare name='ArtDeco00'>
20     <ProtoBody>
21       <Material ambientIntensity='0.25' diffuseColor='0.282435 0.085159 0.134462' emissiveColor='0.0 0.0 0.0' shininess='0.127273' specularColor=
22     </ProtoBody>
23   </ProtoDeclare>
24   <ProtoDeclare name='ArtDeco01'>
25     <ProtoBody>
26       <Material ambientIntensity='0.254777' diffuseColor='0.685208 0.134679 0.332385' emissiveColor='0.0 0.0 0.0' shininess='0.071429' specularCo
27     </ProtoBody>
28   </ProtoDeclare>
29   <ProtoDeclare name='ArtDeco02'>
30     <!-- computed conversion ambientIntensity=1.745282, normalized to 1.0 -->
31     <ProtoBody>
32       <Material ambientIntensity='1.0' diffuseColor='0.536861 0.0529 0.245479' emissiveColor='0.0 0.0 0.0' shininess='0.832432' specularColor='0.
33     </ProtoBody>
34   </ProtoDeclare>
35   <Anchor description='ArtDecoPrototypeExample' parameter='target=_blank' url='"ArtDecoExamplesExcerpt.x3d" "http://X3dGraphics.com/examples/X3dF
36     <Shape>
37       <Appearance>
38         <!-- replace Material node with a corresponding Prototype -->
39         <ProtoInstance containerField='material' name='ArtDeco00'/>
40       </Appearance>
41       <Text string='"ArtDecoPrototypesExcerpt.x3d" is a Materials Prototype declaration file.' "" "For an example scene using these nodes," "cli
42         <FontStyle justify="MIDDLE" "MIDDLE" size='0.8'/>
43       </Text>
44     </Shape>
45   </Anchor>
46 </Scene>
```

# ProtoDeclare editor X3D-Edit

Selecting ProtoDeclare, ProtoInterface or ProtoBody launches the ProtoDeclare interface:



**Edit ProtoDeclare**

Prototype name:

appinfo:

documentation:

**ProtoInterface field definitions**

name	type	accessType	value	appinfo	documentation

+ - ↑ ↓

**Author-assist editing features**

append new ProtoInstance     insert default field values to replace missing appinfo descriptions

append new ExternProtoDeclare     insert new Script node with IS/connect links matching ProtoInterface fields

Accept Discard Help

# Four prototype tooltips

		<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>P</b> ProtoBody	ProtoBody collects ProtoDeclare body nodes. <b>Warning:</b> only the first top-level node and its children are rendered, subsequent nodes (such as Scripts and ROUTEs) will be active but will not be drawn.	
		<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>P</b> ProtoDeclare	ProtoDeclare is a Prototype declaration, defining a new node made up of other node(s). <b>Hint:</b> define field interfaces using the <field> tag, then scene nodes. <b>Hint:</b> initial scene node in a ProtoDeclare body determines this prototype's node type.	
name	[name of the PROTO node being declared NMTOKEN #REQUIRED]	
appinfo	[appinfo type SFString CDATA #IMPLIED] Application information to provide simple description usable as a tooltip, similar to XML Schema appinfo tag.	
documentation	[documentation type SFString CDATA #IMPLIED] Documentation url for further information, similar to XML Schema documentation tag.	
		<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>P</b> ProtoInstance	ProtoInstance creates a copy of a locally or externally defined PROTOtype node. <b>Hint:</b> override default initializations of field values using <fieldValue> tags. <b>Warning:</b> match PROTO node type to local context.	
name	[name of the PROTO node being instanced NMTOKEN #REQUIRED]	
DEF	[DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. <b>Hint:</b> descriptive DEF names improve clarity and help document a model.	
USE	[USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. <b>Hint:</b> USEing other geometry (instead of duplicating nodes) can improve performance. <b>Warning:</b> do NOT include DEF (or any other attribute values) when using a USE attribute!	
containerField	[containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.	
class	[class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.	
		<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>P</b> ProtoInterface	ProtoInterface collects ProtoDeclare field definitions.	
		<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>

	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
 <b>field</b>	<p><b>A field element defines an interface attribute or node.</b></p> <p><b>Hint:</b> first add <code>Script</code>, <code>ProtoDeclare</code> or <code>ExternProtoDeclare</code> before adding a field.</p> <p><b>Hint:</b> put initializing <code>SFNode/MFNode</code> into contained content.</p>
<b>name</b>	<p><b>[name: NMTOKEN #REQUIRED]</b></p> <p>Name of this field variable.</p>
<b>accessType</b>	<p><b>[accessType: (inputOnly outputOnly initializeOnly inputOutput) #REQUIRED]</b></p> <p>Event-model semantics for field set/get capabilities. Hint for VRML 97: inputOnly=eventIn, outputOnly=eventOut, initializeOnly=field, inputOutput=exposedField.</p> <p><b>Warning:</b> inputOutput=exposedField not allowed in VRML 97 Script nodes, use initializeOnly=field for backwards compatibility.</p>
<b>type</b>	<p><b>[type: (select from types list) #REQUIRED]</b></p> <p>Base type of this field variable.</p>
<b>value</b>	<p><b>[value: outputOnly CDATA #IMPLIED]</b></p> <p>Provide default initialization value for this field variable (may be later re-initialized by <code>ProtoInstance</code> <code>fieldValue</code>).</p> <p><b>Hint:</b> <code>SFNode/MFNode</code> are initialized using contained content, instead of this value attribute.</p> <p><b>Hint:</b> required for <code>Script</code> and <code>ProtoDeclare</code>.</p> <p><b>Warning:</b> not allowed for <code>ExternProtoDeclare</code>.</p> <p><b>Warning:</b> not allowed by <code>inputOnly</code> or <code>outputOnly</code> variables.</p>
<b>appinfo</b>	<p><b>[appinfo type SFString CDATA #IMPLIED]</b></p> <p>Application information to provide simple description usable as a tooltip, similar to XML Schema <code>appinfo</code> tag.</p>
<b>documentation</b>	<p><b>[documentation type SFString CDATA #IMPLIED]</b></p> <p>Documentation url for further information, similar to XML Schema <code>documentation</code> tag.</p>
	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>

# <IS> and <connect>

<IS><connect> definitions link field interfaces to internal nodes within the prototype body

These as direct links between outward-facing prototype interface fields and internal fields

- Any initialization or routed input value for the ProtoInterface field definition goes directly into matching internal IS/connect fields
- Any change to a connected internal field is routed out of the prototype, if *accessType*='outputOnly' or *accessType*='inputOutput'

Multiple connections are allowed for each node and for field, both for inputs and for outputs

# <connect>

IS / connect constructs link field interfaces to internal nodes within the prototype declaration

- Each named field IS connected to a prototype field
- Only legal to use within ProtoBody declarations

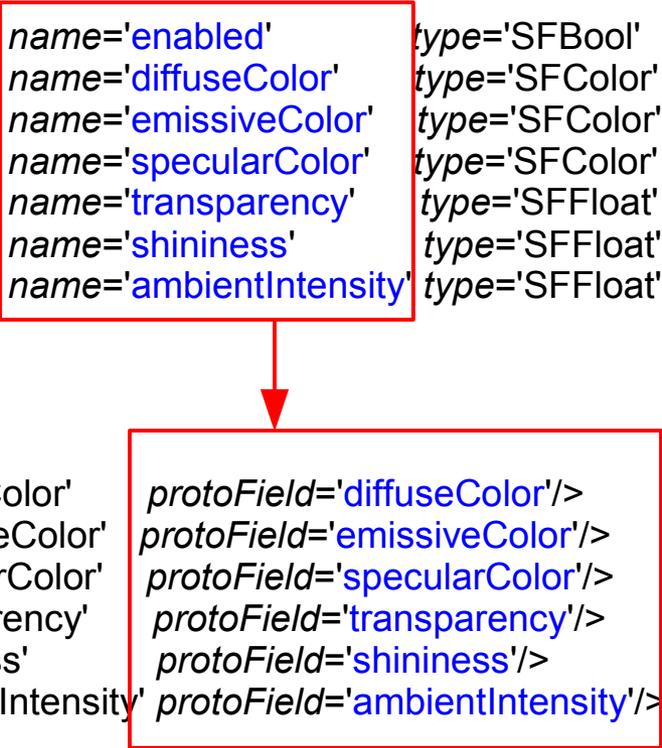
Each <connect> definition provides connection between a given field within local parent node and a corresponding <field> definition in the ProtoInterface

- Each name must match field, interface exactly
- Identical (eponymous) names often best for clarity
- Must also match *type* and *accessType* exactly

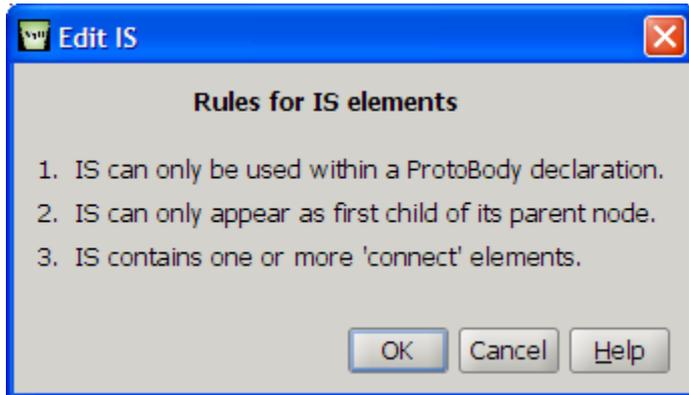
# <IS> and <connect> example

## Prototype interface fields linked to internal fields

```
<ProtoDeclare appinfo='mimic a Material node and modulate fields as an animation effect'  
  name='MaterialModulator'>  
  <ProtoInterface>  
    <field accessType='inputOutput' name='enabled' type='SFBool' value='true'/>  
    <field accessType='inputOutput' name='diffuseColor' type='SFColor' value='0.8 0.8 0.8'/>  
    <field accessType='inputOutput' name='emissiveColor' type='SFColor' value='0 0 0'/>  
    <field accessType='inputOutput' name='specularColor' type='SFColor' value='0 0 0'/>  
    <field accessType='inputOutput' name='transparency' type='SFFloat' value='0.0'/>  
    <field accessType='inputOutput' name='shininess' type='SFFloat' value='0.2'/>  
    <field accessType='inputOutput' name='ambientIntensity' type='SFFloat' value='0.2'/>  
  </ProtoInterface>  
  <ProtoBody>  
    <Material DEF='MaterialNode'>  
      <IS>  
        <connect nodeField='diffuseColor' protoField='diffuseColor'/>  
        <connect nodeField='emissiveColor' protoField='emissiveColor'/>  
        <connect nodeField='specularColor' protoField='specularColor'/>  
        <connect nodeField='transparency' protoField='transparency'/>  
        <connect nodeField='shininess' protoField='shininess'/>  
        <connect nodeField='ambientIntensity' protoField='ambientIntensity'/>  
      </IS>  
    </Material> <!-- etc. -->  
  </ProtoBody>  
</ProtoDeclare>
```

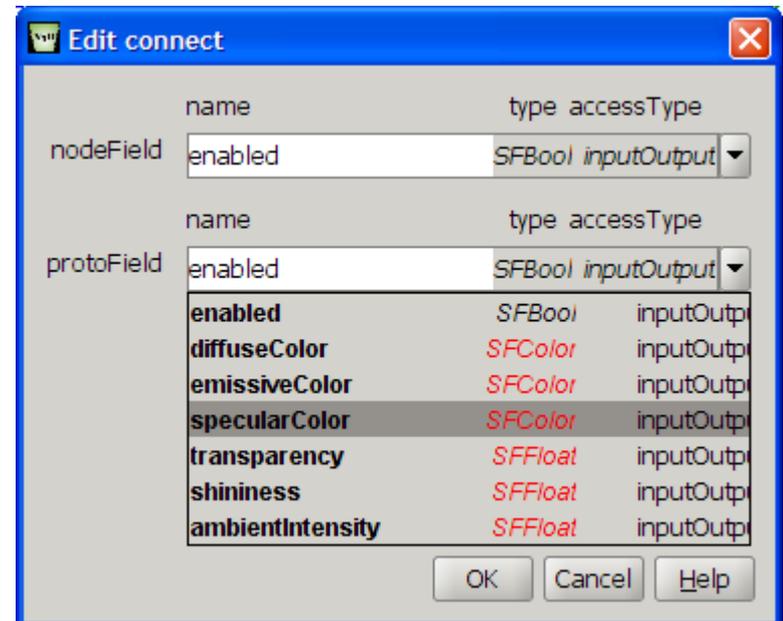
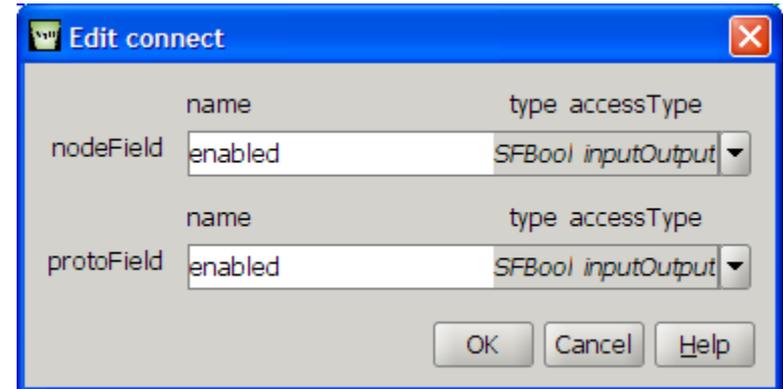


# IS / connect in X3D-Edit



<IS> editor is simple

<connect> editor prompts author to connect proper type and accessType between parent-node and prototype fields



	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>IS</b>	<p><b>IS</b> connects Prototype interface fields to node fields inside ProtoDeclare definitions. Add one or more connect tags to define each pair of Prototype field connections.</p> <p><b>Warning:</b> IS tag only allowed within ProtoDeclare body definitions.</p> <p><b>Hint:</b> IS tag precedes any Metadata tag, which precedes any other children tags.</p>
	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>

	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>connect</b>	<p><b>connect</b> tags define each Prototype field connection within ProtoDeclare definitions.</p> <p><b>Warning:</b> IS/connect tags are only allowed within ProtoDeclare body definitions.</p>
nodeField	<p><b>[nodeField: NMTOKEN #REQUIRED]</b>  Name of field in this node connecting to parent ProtoDeclare field definition.  <b>Hint:</b> use multiple connect tags for multiple fan-in/fan-out.</p>
protoField	<p><b>[protoField: NMTOKEN #REQUIRED]</b>  Name of parent ProtoDeclare field definition connecting to field in this node.  <b>Hint:</b> use multiple connect tags for multiple fan-in/fan-out.</p>
	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>

# Connecting an embedded Script 1

A common design goal: create a Prototype that is modified version of specific node

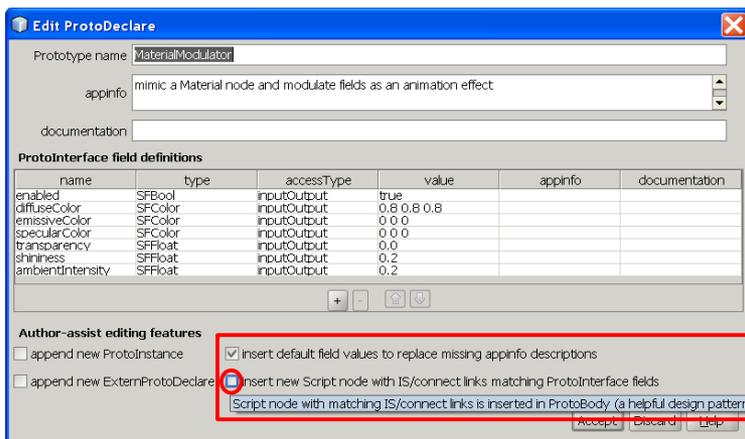
Example:

- Prototype name='NewMaterial'
- ProtoInterface holds definitions for all original fields plus possibly some additional fields
- ProtoBody initial node is essential: e.g. Material, fully linked by IS/connect definitions for each field
- Next (nonrendered) node is modifying Script, also holding IS/connect field definitions plus connection to Material (via ROUTE or DEF/USE in a field)

# Connecting an embedded Script 2

## X3D-Edit can insert Script if fields are defined

- May eventually add support for full design pattern



autogenerated X3D

<ProtoBody>

...

<Script DEF='MaterialModulatorScript'>

<field accessType='inputOutput' name='enabled' type='SFBool'/>

<field accessType='inputOutput' name='diffuseColor' type='SFColor'/>

<field accessType='inputOutput' name='emissiveColor' type='SFColor'/>

<field accessType='inputOutput' name='specularColor' type='SFColor'/>

<field accessType='inputOutput' name='transparency' type='SFFloat'/>

<field accessType='inputOutput' name='shininess' type='SFFloat'/>

<field accessType='inputOutput' name='ambientIntensity' type='SFFloat'/>

<IS>

<connect nodeField='enabled' protoField='enabled'/>

<connect nodeField='diffuseColor' protoField='diffuseColor'/>

<connect nodeField='emissiveColor' protoField='emissiveColor'/>

<connect nodeField='specularColor' protoField='specularColor'/>

<connect nodeField='transparency' protoField='transparency'/>

<connect nodeField='shininess' protoField='shininess'/>

<connect nodeField='ambientIntensity' protoField='ambientIntensity'/>

</IS>

</Script>

</ProtoBody>

# ExternProtoDeclare

ExternProtoDeclare references an individual ProtoDeclare definition in an external scene

- Allows single “master” definition of a prototype, avoids versionitis from cut/paste redistributions
- Multiple prototype nodes require multiple ExternProtoDeclare statements

Includes <field> definitions matching interface signature of the original prototype

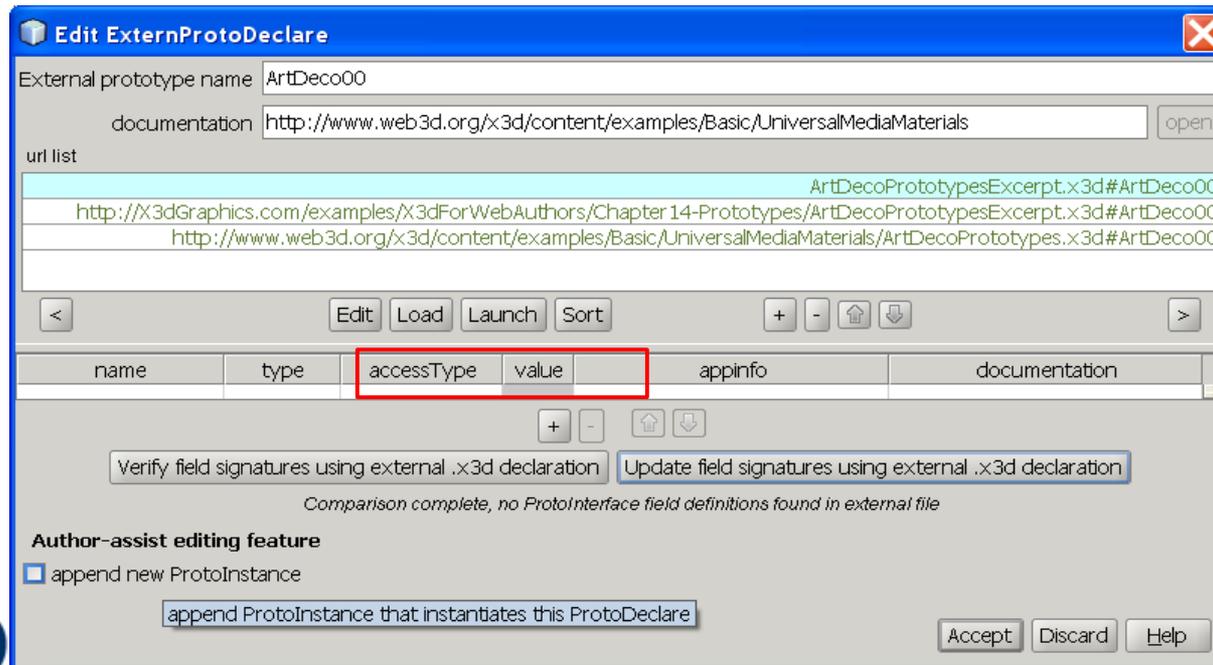
- Minus initial values, so that conflicts are avoided
- Allows X3D browser to “understand” new nodes and create proper scene graph when loading

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
3 <X3D profile='Immersive' version='3.0' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/sp
4 <head>
5 <meta content='ArtDecoExamplesExcerpt.x3d' name='title' />
6 <meta content='Example ExternProtoDeclare/ProtoInstance usage of X3D/VRML materials, originally converted from SGI&apos;s Open Inventor materia
7 <meta content='David Roussel' name='creator' />
8 <meta content='James Harney, Don Brutzman NPS' name='translator' />
9 <meta content='7 April 2002' name='created' />
10 <meta content='4 August 2008' name='modified' />
11 <meta content='http://vrmlstuff.free.fr/materials' name='reference' />
12 <meta content='Universal Media Material Library' name='subject' />
13 <meta content='http://www.web3d.org/x3d/content/examples/Basic/UniversalMediaMaterials/ArtDecoExamples.x3d' name='reference' />
14 <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ArtDecoExamplesExcerpt.x3d' name='identifier' />
15 <meta content='Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html' name='generator' />
16 <meta content='../license.html' name='license' />
17 </head>
18 <Scene>
19 <ExternProtoDeclare name='ArtDeco00' url='ArtDecoPrototypesExcerpt.x3d#ArtDeco00'
20 "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ArtDecoPrototypesExcerpt.x3d#ArtDeco00" "http://www.web3d.org/x3d/co
21 <ExternProtoDeclare name='ArtDeco01' url='ArtDecoPrototypesExcerpt.x3d#ArtDeco01'
22 "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ArtDecoPrototypesExcerpt.x3d#ArtDeco01" "http://www.web3d.org/x3d/co
23 <ExternProtoDeclare name='ArtDeco02' url='ArtDecoPrototypesExcerpt.x3d#ArtDeco02'
24 "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ArtDecoPrototypesExcerpt.x3d#ArtDeco02" "http://www.web3d.org/x3d/co
25 <Group>
26 <NavigationInfo headlight='false' />
27 <Viewpoint DEF='Front' description='Front' fieldOfView='0.785398' position='0.0 0.0 12.0' />
28 <Viewpoint DEF='PersRight' description='Low Right' fieldOfView='0.785398' orientation='0.74291 0.30772 0.59447 1.2171' position='6.9282 -6.92
29 <Viewpoint DEF='PersLeft' description='Low Left' fieldOfView='0.785398' orientation='0.74291 -0.30772 -0.59447 1.2171' position='-6.9282 -6.9
30 <Viewpoint DEF='Back' description='Back' fieldOfView='0.785398' orientation='0.0 1.0 0.0 3.1416' position='0.0 0.0 -12.0' />
31 <Transform DEF='Close_travel'>
32 <PositionInterpolator DEF='Close_Mover' key='0.0 0.25 0.5 0.75 1.0' keyValue='0.0 2.5 0.0 0.0 0.0 0.0 0.0 -2.5 0.0 0.0 0.0 0.0 0.0 2.5 0.0'
33 <TimeSensor DEF='Close_Time' cycleInterval='12.0' loop='true' />
34 <Viewpoint DEF='Close' description='Close Front' fieldOfView='0.785398' position='0.0 0.0 6.0' />
35 </Transform>
36 <DirectionalLight direction='1.0 -1.0 -1.0' />
37 <DirectionalLight direction='0.0 1.0 -0.5' intensity='0.5' />
38 <Anchor description='Back to front view' url='"#Front"'>
39 <Transform translation='0.0 0.0 -0.5'>
40 <Inline url='gridBack.x3d' "http://www.web3d.org/x3d/content/examples/Basic/UniversalMediaMaterials/gridBack.x3d" />
41 </Transform>
42 </Anchor>
43 <Viewpoint DEF='View00' description='ArtDeco00' fieldOfView='0.785398' position='-3.75 3.75 3.0' />
44 <Transform translation='-3.75 3.75 0.0'>
45 <Anchor description='ArtDeco00 view' url='"#View00"'>
46 <Shape>
47 <Appearance>
```

# ExternProtoDeclare editor X3D-Edit

## ExternProtoDeclare editor for multiple url values

- Note #ProtoName appended to each filename
- Can edit, locally load, or launch each address
- Can sort url list (relative, .x3d before online, .wrl)



# *appinfo*, *documentation* attributes

The *appinfo* and *documentation* attributes accompany ProtoDeclare, ExternProtoDeclare and field definitions

- *appinfo* holds a simple summary or tooltip
- *documentation* holds a url to further information

These match identical constructs in XML Schema

- Allowing tools to further support authoring, editing
- Allowing authors to properly document new nodes

These are important to use, and help long-term extensibility of your work and X3D itself

# ProtoInstance

Finally you can make copies of your new node:  
create Prototype instances using ProtoInstance

- Must be preceded by either ProtoDeclare or ExternProtoDeclare with same name
- Otherwise a run-time error results for end user

Nevertheless simple to invoke and instantiate:

```
<ProtoInstance name='ArtDeco00'/>
```

Can override default initialization values for fields

- This is how a prototype is customized upon creation
- ```
<fieldValue name='someField' value='someValue'/>
```
- Can also initialize child nodes, if any

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
3 <X3D profile='Immersive' version='3.0' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/sp
4 <head>
5 <meta content='ArtDecoExamplesExcerpt.x3d' name='title' />
6 <meta content='Example ExternProtoDeclare/ProtoInstance usage of X3D/VRML materials, originally converted from SGI&apos;s Open Inventor materia
7 <meta content='David Roussel' name='creator' />
8 <meta content='James Harney, Don Brutzman NPS' name='translator' />
9 <meta content='7 April 2002' name='created' />
10 <meta content='4 August 2008' name='modified' />
11 <meta content='http://vrmlstuff.free.fr/materials' name='reference' />
12 <meta content='Universal Media Material Library' name='subject' />
13 <meta content='http://www.web3d.org/x3d/content/examples/Basic/UniversalMediaMaterials/ArtDecoExamples.x3d' name='reference' />
14 <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ArtDecoExamplesExcerpt.x3d' name='identifier' />
15 <meta content='Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html' name='generator' />
16 <meta content='../license.html' name='license' />
17 </head>
18 <Scene>
19 <ExternProtoDeclare name='ArtDeco00' url='ArtDecoPrototypesExcerpt.x3d#ArtDeco00' "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-
20 <ExternProtoDeclare name='ArtDeco01' url='ArtDecoPrototypesExcerpt.x3d#ArtDeco01' "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-
21 <ExternProtoDeclare name='ArtDeco02' url='ArtDecoPrototypesExcerpt.x3d#ArtDeco02' "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-
22 <Group>
23 <NavigationInfo headlight='false' />
24 <Viewpoint DEF='Front' description='Front' fieldOfView='0.785398' position='0.0 0.0 12.0' />
25 <Viewpoint DEF='PersRight' description='Low Right' fieldOfView='0.785398' orientation='0.74291 0.30772 0.59447 1.2171' position='6.9282 -6.92
26 <Viewpoint DEF='PersLeft' description='Low Left' fieldOfView='0.785398' orientation='0.74291 -0.30772 -0.59447 1.2171' position='-6.9282 -6.9
27 <Viewpoint DEF='Back' description='Back' fieldOfView='0.785398' orientation='0.0 1.0 0.0 3.1416' position='0.0 0.0 -12.0' />
28 <Transform>
29 <DirectionalLight direction='1.0 -1.0 -1.0' />
30 <DirectionalLight direction='0.0 1.0 -0.5' intensity='0.5' />
31 <Anchor description='Back to front view' url=' "#Front"'>
32 <Transform translation='0.0 0.0 -0.5'>
33 <Inline url=' "gridBack.x3d" "http://www.web3d.org/x3d/content/examples/Basic/UniversalMediaMaterials/gridBack.x3d" />
34 </Transform>
35 </Anchor>
36 <Viewpoint DEF='View00' description='ArtDeco00' fieldOfView='0.785398' position='-3.75 3.75 0.0'>
37 <Transform translation='-3.75 3.75 0.0'>
38 <Anchor description='ArtDeco00 view' url=' "#View00"'>
39 <Shape>
40 <Appearance>
41 <ProtoInstance containerField='material' name='ArtDeco00' />
42 </Appearance>
43 <Sphere DEF='Ball' radius='0.5' />
44 </Shape>
45 </Anchor>
46 <Transform translation='0.0 0.3 0.5'>

```

ArtDecoExamplesExcerpt.x3d

**Edit ProtoInstance**

containerField: material (checked)

DEF: [ ]

USE: [ ]

Referenced Prototype: ArtDeco00

fieldValue initializations

A	B	C	D	E

[+] [-]

OK Cancel Help

# *containerField* considerations

*containerField* is how the field name for a node is provided, relative to the node's parent

- Usually not needed since default matches most common case: *containerField* = 'children'
- ClassicVRML syntax is different, more verbose
- As ever, functionality is identical

```
<!-- Rendered geometry follows prototype declaration -->
<Shape>
  <Sphere/>
  <Appearance>
    <ProtoInstance containerField='material'
                    name='MaterialModulator'>
      <fieldValue name='enabled' value='true!'/>
      <fieldValue name='diffuseColor' value='0.5 0.1 0.1!'/>
    </ProtoInstance>
  </Appearance>
</Shape>
```

```
# Rendered geometry follows prototype declaration
Shape {
  geometry Sphere {
  }
  appearance Appearance {
    material MaterialModulator {
      enabled TRUE
      diffuseColor 0.5 0.1 0.1
    }
  }
}
```

# fieldValue initializations 1

fieldValue name must match; initialization values must match the type specified in declaration

- Otherwise a run-time error results for end user
- Take special care to check correctness, avoid errors

To initialize simple types: use *value* parameter

```
<ProtoInstance name='MaterialModulator'  
  containerField='material'  
  <fieldValue name='enabled' value='true'/>  
  <fieldValue name='diffuseColor' value='0.5 0.1 0.1'/>  
</ProtoInstance>
```

# fieldValue initializations 2

To initialize SFNode or MFNode types, use contained nodes within the fieldValue element:

```
<ProtoInstance name='SomethingNew'>  
  <fieldValue name='newSFNodeField'>  
    <!-- initialization node goes here -->  
  </fieldValue>  
</ProtoInstance>
```

As might be expected, fieldValue initializations are only allowed for fields with *accessType* of initializeOnly or inputOutput

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
3 <X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/spe
4 <head>
5   <meta content='MaterialModulator.x3d' name='title' />
6   <meta content='Mimic a Material node and modulate fields as an animation effect' name='description' />
7   <meta content='Don Brutzman' name='creator' />
8   <meta content='10 March 2008' name='created' />
9   <meta content='3 August 2008' name='modified' />
10  <meta content='X3D prototype requiring Script inputOutput fields' name='subject' />
11  <meta content='MaterialModulator.png' name='image' />
12  <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/MaterialModulator.x3d' name='identifier' />
13  <meta content='X3D-Edit 3.2, https://savage.nps.edu/X3D-Edit' name='generator' />
14  <meta content='.././././license.html' name='license' />
15 </head>
16 <Scene>
17   <ProtoDeclare appinfo='mimic a Material node and modulate fields as an animation effect' name='MaterialModulator'>
18     <ProtoInterface>
19       <field accessType='inputOutput' name='enabled' type='SFBool' value='true' />
20       <field accessType='inputOutput' name='diffuseColor' type='SFCOLOR' value='0.8 0.8 0.8' />
21       <field accessType='inputOutput' name='emissiveColor' type='SFCOLOR' value='0 0 0' />
22       <field accessType='inputOutput' name='specularColor' type='SFCOLOR' value='0 0 0' />
23       <field accessType='inputOutput' name='transparency' type='SFFloat' value='0.0' />
24       <field accessType='inputOutput' name='shininess' type='SFFloat' value='0.2' />
25       <field accessType='inputOutput' name='ambientIntensity' type='SFFloat' value='0.2' />
26     </ProtoInterface>
27     <ProtoBody>
28       <Material DEF='MaterialNode'>
29         <IS>
30           <connect nodeField='diffuseColor' protoField='diffuseColor' />
31           <connect nodeField='emissiveColor' protoField='emissiveColor' />
32           <connect nodeField='specularColor' protoField='specularColor' />
33           <connect nodeField='transparency' protoField='transparency' />
34           <connect nodeField='shininess' protoField='shininess' />
35           <connect nodeField='ambientIntensity' protoField='ambientIntensity' />
36         </IS>
37       </Material>
38       <Script DEF='MaterialModulatorScript' directOutput='true'>
39         <field accessType='inputOutput' name='enabled' type='SFBool' />
40         <field accessType='inputOutput' name='diffuseColor' type='SFCOLOR' />
41         <field accessType='outputOnly' name='newColor' type='SFCOLOR' value='0 0 0' />
42         <field accessType='inputOnly' name='clockTrigger' type='SFTime' />
43         <IS>
44           <connect nodeField='enabled' protoField='enabled' />
45           <connect nodeField='diffuseColor' protoField='diffuseColor' />
46         </IS>
47       </Script>
48     </ProtoBody>
49   </ProtoDeclare>
50   <Material DEF='MaterialNode'>
51     <IS>
52       <connect nodeField='diffuseColor' protoField='diffuseColor' />
53       <connect nodeField='emissiveColor' protoField='emissiveColor' />
54       <connect nodeField='specularColor' protoField='specularColor' />
55       <connect nodeField='transparency' protoField='transparency' />
56       <connect nodeField='shininess' protoField='shininess' />
57       <connect nodeField='ambientIntensity' protoField='ambientIntensity' />
58     </IS>
59   </Material>
60   <Script DEF='MaterialModulatorScript' directOutput='true'>
61     <field accessType='inputOutput' name='enabled' type='SFBool' />
62     <field accessType='inputOutput' name='diffuseColor' type='SFCOLOR' />
63     <field accessType='outputOnly' name='newColor' type='SFCOLOR' value='0 0 0' />
64     <field accessType='inputOnly' name='clockTrigger' type='SFTime' />
65     <IS>
66       <connect nodeField='enabled' protoField='enabled' />
67       <connect nodeField='diffuseColor' protoField='diffuseColor' />
68     </IS>
69   </Script>
70 </Scene>

```

## MaterialModulator.x3d part 1

**Edit ProtoDeclare**

Prototype name:

appinfo:

documentation:

**ProtoInterface field definitions**

name	type	accessType	value	appinfo	documentation
enabled	SFBool	inputOutput	true		
diffuseColor	SFCOLOR	inputOutput	0.8 0.8 0.8		
emissiveColor	SFCOLOR	inputOutput	0 0 0		
specularColor	SFCOLOR	inputOutput	0 0 0		
transparency	SFFloat	inputOutput	0.0		
shininess	SFFloat	inputOutput	0.2		
ambientIntensity	SFFloat	inputOutput	0.2		

Author-assist editing features

append new ProtoInstance  insert default field values to replace missing appinfo descriptions

append new ExternProtoDeclare  insert new Script node with IS/connect links matching ProtoInterface fields

Accept Discard Help

```

37 </Material>
38 <Script DEF='MaterialModulatorScript' directOutput='true'>
39   <field accessType='inputOutput' name='enabled' type='SFBool' />
40   <field accessType='inputOutput' name='diffuseColor' type='SFColor' />
41   <field accessType='outputOnly' name='newColor' type='SFColor' value='0 0 0' />
42   <field accessType='inputOnly' name='clockTrigger' type='SFTime' />
43   <IS>
44     <connect nodeField='enabled' protoField='enabled' />
45     <connect nodeField='diffuseColor' protoField='diffuseColor' />
46   </IS>
47   <![CDATA[
48   ecmascript:
49
50   function initialize ()
51   {
52     newColor = diffuseColor; // start with correct color
53   }
54
55   function clockTrigger (timeValue)
56   {
57     if (!enabled) return;
58     red   = newColor.r;
59     green = newColor.g;
60     blue  = newColor.b;
61
62     // note different modulation rates for each color component, % is modulus operator
63     newColor = new SFColor ((red + 0.02) % 1, (green + 0.03) % 1, (blue + 0.04) % 1);
64     Browser.print ('diffuseColor=(' + red + ', ' + green + ', ' + blue + ') newColor=' + newColor);
65   }
66 ]>
67 </Script>
68 <ROUTE fromField='newColor' fromNode='MaterialModulatorScript' toField='diffuseColor' />
69 <TimeSensor DEF='ModulationClock' cycleInterval='0.05' loop='true' />
70 <ROUTE fromField='cycleTime' fromNode='ModulationClock' toField='clockTrigger' toProto='MaterialModulator' />
71 </ProtoBody>
72 </ProtoDeclare>
73 <!-- Rendered geometry follows prototype declaration -->
74 <Shape>
75   <Sphere />
76   <Appearance>
77     <ProtoInstance containerField='material' name='MaterialModulator'>
78       <fieldValue name='enabled' value='true' />
79       <fieldValue name='diffuseColor' value='0.5 0.1 0.1' />
80     </ProtoInstance>
81   </Appearance>
82 </Shape>
83 </Scene>
84 </X3D>

```

## MaterialModulator.x3d part 2

**Edit ProtoInstance**

containerField:  DEF

material  USE

Referred Prototype: MaterialModulator

fieldValue initializations

override	name	type	accessType	value
<input checked="" type="checkbox"/>	enabled	SFBool	inputOutput	true
<input checked="" type="checkbox"/>	diffuseColor	SFColor	inputOutput	0.5 0.1 0.1
<input type="checkbox"/>	emissiveColor	SFColor	inputOutput	0 0 0
<input type="checkbox"/>	specularColor	SFColor	inputOutput	0 0 0
<input type="checkbox"/>	transparency	SFFloat	inputOutput	0.0
<input type="checkbox"/>	shininess	SFFloat	inputOutput	0.2
<input type="checkbox"/>	ambientIntensity	SFFloat	inputOutput	0.2

+ -

OK Cancel Help

**Edit fieldValue**

ProtoDeclare name: MaterialModulator

ProtoInstance fieldValue:

name: enabled

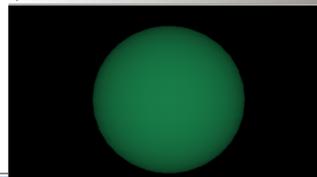
value: true

OK Cancel Help

X3D Viewer

X3D Viewer

X3D Viewer



<b>P ProtoInstance</b>	<b>ProtoInstance creates a copy of a locally or externally defined PROTOtype node.</b> <b>Hint: override default initializations of field values using &lt;fieldValue&gt; tags.</b> <b>Warning: match PROTO node type to local context.</b>
name	<b>[name of the PROTO node being instanced NMTOKEN #REQUIRED]</b>
DEF	<b>[DEF ID #IMPLIED]</b> DEF defines a unique ID name for this node, referencable by other nodes. <b>Hint:</b> descriptive DEF names improve clarity and help document a model.
USE	<b>[USE IDREF #IMPLIED]</b> USE means reuse an already DEF-ed node ID, ignoring <code>_all_</code> other attributes and children. <b>Hint:</b> USEing other geometry (instead of duplicating nodes) can improve performance. <b>Warning:</b> do NOT include DEF (or any other attribute values) when using a USE attribute!
containerField	<b>[containerField: NMTOKEN "children"]</b> containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.
class	<b>[class CDATA #IMPLIED]</b> class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.

	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>fieldValue</b>	<b>A fieldValue element is used to re-initialize default field values in ProtoInstances. Field names must be already defined in ProtoDeclare or ExternProtoDeclare.</b> <b>Hint: put initializing SFNode/MFNode into fieldValue's contained content.</b>
name	<b>[name: NMTOKEN #REQUIRED]</b> Name of this field (already defined in ProtoDeclare or ExternProtoDeclare).
value	<b>[value: outputOnly CDATA #IMPLIED]</b> Initial value for this field (overrides default initialization value in ProtoDeclare or ExternProtoDeclare). <b>Hint:</b> initialize SFNode/MFNode using contained content instead.
	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>

# Advanced Examples

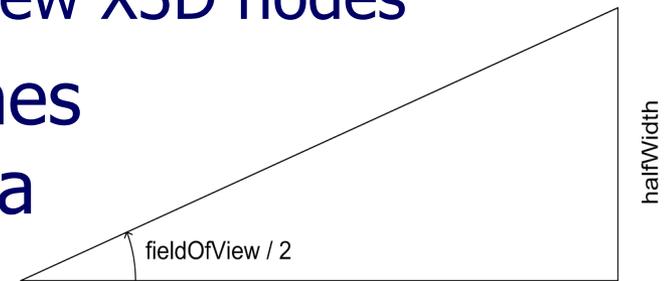
# Detailed example: ViewFrustum

ViewFrustum is a helpful visualization prototype

- Prototypes simplify creation of new X3D nodes

Shows near and far clipping planes that truncate the viewable area

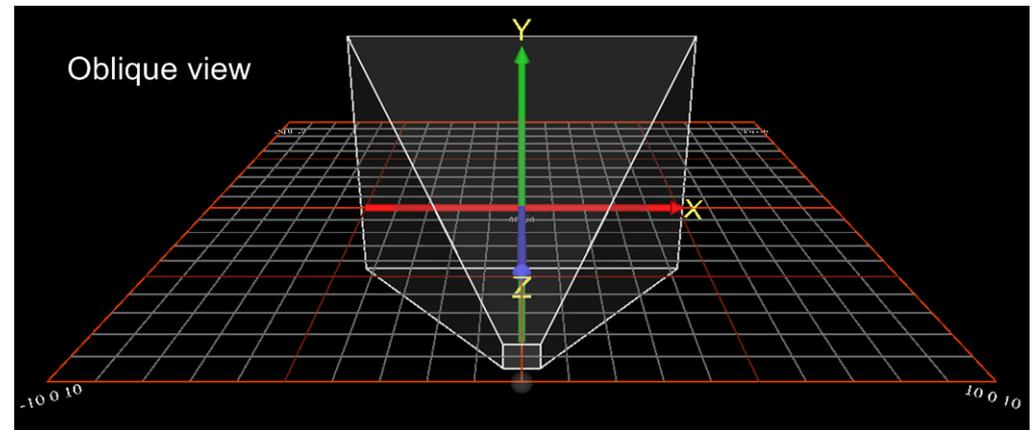
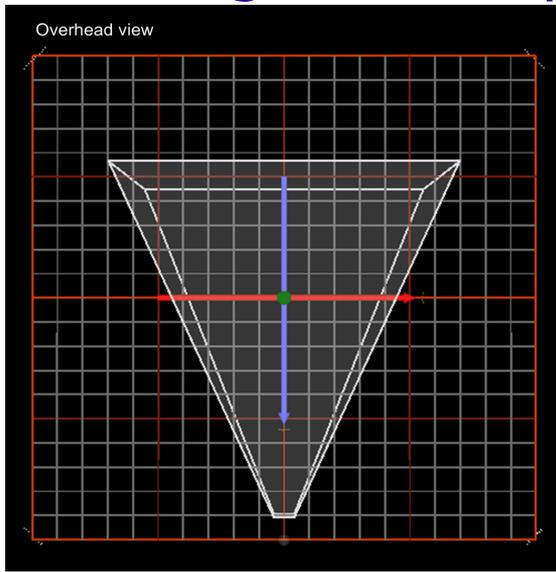
- Depends on Viewpoint and NavigationInfo parameters



Near clipping plane distance =  $avatarSize[0]$

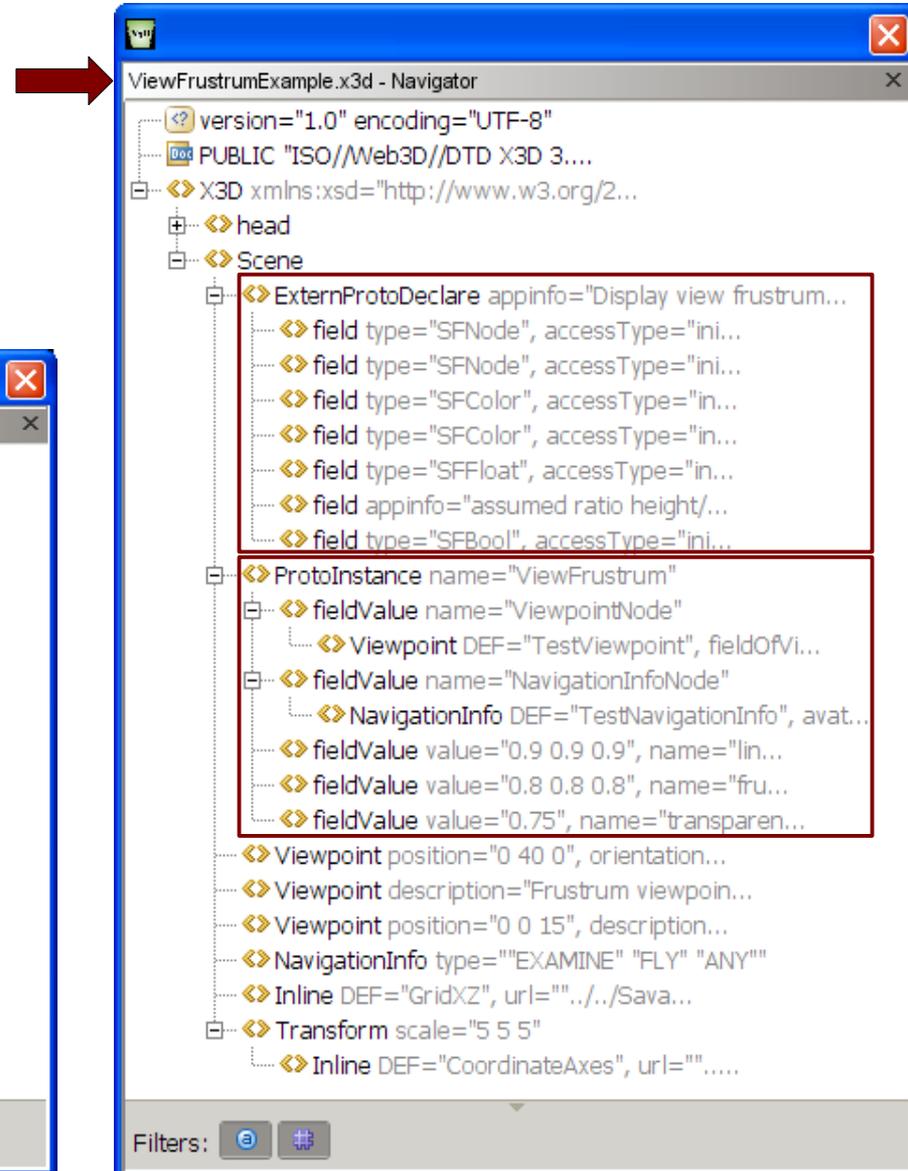
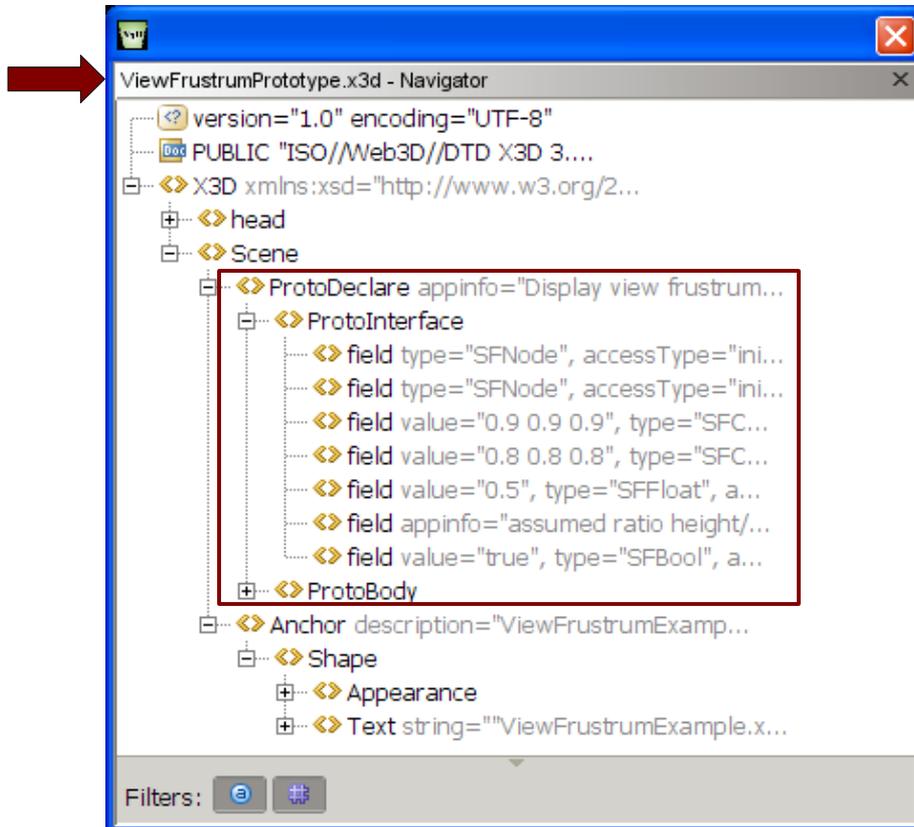
Far clipping plane distance =  $visibilityLimit$

$$\begin{aligned} \text{nearHalfWidth} &= \tan(\text{fieldOfView} / 2) * \text{avatarSize}[0]; \\ \text{farHalfWidth} &= \tan(\text{fieldOfView} / 2) * \text{visibilityLimit}; \end{aligned}$$



# ViewFrustrum prototype, example

Good practice: make two separate files to simplify ExternProtoDeclare reuse



# Prototype features of interest

Highlighted ProtoDeclare, ExternProtoDeclare, ProtoInstance and Script show:

- Using initialize() method to setup geometry nodes
- Usage of IS/connect for direct node inspection
- Usage of event-passing via ROUTE when changing Extrusion, which doesn't support direct modification
- Matching *type* and *accessType*, toString() function
- External script code, accessing node fields
- Duplicate *url* addresses, local and remote
- Browser.println statements, silencable by *trace* field
- Internal var declarations, Javascript Math library

```

19 <Scene>
20 <ProtoDeclare appinfo='Display view frustrum associated with a given pair of Viewpoint NavigationInfo nodes' name='ViewFrustrum'>
21 <ProtoInterface>
22 <field accessType='initializeOnly' name='ViewpointNode' type='SFNode'>
23 <!-- NULL node -->
24 </field>
25 <field accessType='initializeOnly' name='NavigationInfoNode' type='SFNode'>
26 <!-- NULL node -->
27 </field>
28 <field accessType='inputOutput' name='lineColor' type='SFCOLOR' value='0.9 0.9 0.9'>
29 <field accessType='inputOutput' name='frustrumColor' type='SFCOLOR' value='0.8 0.8 0.8'>
30 <field accessType='inputOutput' name='transparency' type='SFFloat' value='0.5'>
31 <field accessType='inputOutput' appinfo='assumed ratio height/width' name='aspectRatio' type='SFFloat' value='0.75'>
32 <field accessType='initializeOnly' name='trace' type='SFBool' value='true'>
33 </ProtoInterface>
34 <ProtoBody>
35 <Transform DEF='PositionTransform' rotation='0 1 0 3.14159'>
36 <Transform DEF='OrientationTransform'>
37 <Shape>
38 <IndexedLineSet DEF='FrustrumLines' coordIndex='0 1 2 3 0 -1 4 5 6 7 4 -1 0 4 -1 1 5 -1 2 6 -1 3 7 -1'>
39 <Coordinate DEF='FrustrumCoordinate' point='0 0'>
40 </IndexedLineSet>
41 <Appearance>
42 <Material>
43 <IS>
44 <connect nodeField='emissiveColor' protoField='lineColor'>
45 </IS>
46 </Material>
47 </Appearance>
48 </Shape>
49 <Shape>
50 <Extrusion DEF='FrustrumExtrusion'>
51 <Appearance DEF='FrustrumAppearance'>
52 <Material>
53 <IS>
54 <connect nodeField='diffuseColor' protoField='frustrumColor'>
55 <connect nodeField='transparency' protoField='transparency'>
56 </IS>
57 </Material>
58 </Appearance>
59 </Shape>
60 <Shape>
61 <Sphere radius='0.25'>
62 <Appearance USE='FrustrumAppearance'>
63 </Shape>
64 </Transform>
65 </Transform>

```

field definitions

Coordinate points  
for outline,  
need initialization

Extrusion for frustrum  
polygons,  
need initialization

Small Sphere shows  
Viewpoint position

```

63 </Shape>
64 </Transform>
65 </Transform>
66 <Script DEF='GeometryComputationScript' directOutput='true' url='"ViewFrustrumScript.js"' "http://X3dGraphics.com/examples/X3dForWebAuthors/
67 <field accessType='initializeOnly' name='ViewpointNode' type='SFNode'>/>
68 <field accessType='initializeOnly' name='NavigationInfoNode' type='SFNode'>/>
69 <field accessType='initializeOnly' name='FrustrumCoordinate' type='SFNode'>
70 <Coordinate USE='FrustrumCoordinate'>/>
71 </field>
72 <field accessType='initializeOnly' name='FrustrumExtrusion' type='SFNode'>
73 <Extrusion USE='FrustrumExtrusion'>/>
74 </field>
75 <field accessType='inputOnly' name='recompute' type='SFBool'>/>
76 <field accessType='inputOutput' appinfo='assumed ratio height/width' name='aspectRatio' type='SFFloat'>/>
77 <field accessType='outputOnly' name='position_changed' type='SFVec3f'>/>
78 <field accessType='outputOnly' name='orientation_changed' type='SFRotation'>/>
79 <field accessType='outputOnly' name='spine_changed' type='MFVec3f'>/>
80 <field accessType='outputOnly' name='scale_changed' type='MFVec2f'>/>
81 <field accessType='outputOnly' name='point_changed' type='MFVec3f'>/>
82 <field accessType='initializeOnly' name='trace' type='SFBool'>/>
83 <IS>
84 <connect nodeField='ViewpointNode' protoField='ViewpointNode'>/>
85 <connect nodeField='NavigationInfoNode' protoField='NavigationInfoNode'>/>
86 <connect nodeField='aspectRatio' protoField='aspectRatio'>/>
87 <connect nodeField='trace' protoField='trace'>/>
88 </IS>
89 </Script>
90 <ROUTE fromField='position_changed' fromNode='GeometryComputationScript' toField='translation' toNode='PositionTransform'>/>
91 <ROUTE fromField='orientation_changed' fromNode='GeometryComputationScript' toField='rotation' toNode='OrientationTransform'>/>
92 <ROUTE fromField='spine_changed' fromNode='GeometryComputationScript' toField='set_spine' toNode='FrustrumExtrusion'>/>
93 <ROUTE fromField='scale_changed' fromNode='GeometryComputationScript' toField='set_scale' toNode='FrustrumExtrusion'>/>
94 <ROUTE fromField='point_changed' fromNode='GeometryComputationScript' toField='point' toNode='FrustrumCoordinate'>/>
95 </ProtoBody>
96 </ProtoDeclare>
97 <!-- Example use is in separate scene -->
98 <Anchor description='ViewFrustrumExample' parameter='target=_blank' url='"ViewFrustrumExample.x3d"' "http://X3dGraphics.com/examples/X3dForWebAu
99 <Shape>
100 <Appearance>
101 <Material diffuseColor='0.8 0.4 0'>/>
102 </Appearance>
103 <Text string='"ViewFrustrumExample.x3d" "is a Prototype declaration file." "" "For an example scene using this node," "click this text and
104 <FontStyle justify="MIDDLE" "MIDDLE" size="0.8'>/>
105 </Text>
106 </Shape>
107 </Anchor>
108 </Scene>
109 </X3D>

```

Match  
ProtoInterface  
field definitions

Output fields for  
ROUTE links

IS/connect links  
match field definitions

ROUTE  
links

User selects Text message  
to launch example scene

```
ViewFrustrumScript.js - Editor
ViewFrustrumScript.js x
function initialize ()
{
    var scriptName = 'ViewFrustrumScript';

    if ((ViewpointNode == null) || (NavigationInfoNode == null))
    {
        Browser.println ('[' + scriptName + '] ' + 'Viewpoint and/or NavigationInfo undefined, no ViewFrustrum drawn');
        return;
    }

    if (trace) Browser.println ('[' + scriptName + '] ' + 'input ' +
        '<Viewpoint position=' + ViewpointNode.position.toString() + '"' +
        ' orientation=' + ViewpointNode.orientation.toString() + '"' +
        ' fieldOfView=' + ViewpointNode.fieldOfView.toString() + '"' + '>');
    position_changed = ViewpointNode.position;
    orientation_changed = ViewpointNode.orientation_changed;

    if (trace) Browser.println ('[' + scriptName + '] ' + 'input ' +
        '<NavigationInfo avatarSize=' + NavigationInfoNode.avatarSize.toString() + '"' +
        ' visibilityLimit=' + NavigationInfoNode.visibilityLimit.toString() + '"' + '>');
    var nearClipPlaneDistance = NavigationInfoNode.avatarSize[0];
    var farClipPlaneDistance = NavigationInfoNode.visibilityLimit;
    if (farClipPlaneDistance == 0) farClipPlaneDistance = 10000.0;

    var nearHalfWidth = Math.tan(ViewpointNode.fieldOfView / 2.0) * nearClipPlaneDistance;
    var farHalfWidth = Math.tan(ViewpointNode.fieldOfView / 2.0) * farClipPlaneDistance;

    spine_changed = new MFVec3f (new SFVec3f (0.0, 0.0, nearClipPlaneDistance),
        new SFVec3f (0.0, 0.0, farClipPlaneDistance));
    scale_changed = new MFVec2f (new SFVec2f (nearHalfWidth, aspectRatio * nearHalfWidth),
        new SFVec2f (farHalfWidth, aspectRatio * farHalfWidth));

    if (trace) Browser.println ('[' + scriptName + '] ' + 'output ' +
        '<Extrusion DEF="FrustrumExtrusion" ' +
        ' spine=' + spine_changed.toString() + '"' +
        ' scale=' + scale_changed.toString() + '"' + '>'); // default crossSection used

    point_changed = new MFVec3f (
        new SFVec3f ( nearHalfWidth, aspectRatio * nearHalfWidth, nearClipPlaneDistance),
        new SFVec3f ( nearHalfWidth, -aspectRatio * nearHalfWidth, nearClipPlaneDistance),
        new SFVec3f (-nearHalfWidth, -aspectRatio * nearHalfWidth, nearClipPlaneDistance),
        new SFVec3f (-nearHalfWidth, aspectRatio * nearHalfWidth, nearClipPlaneDistance),
        new SFVec3f ( farHalfWidth, aspectRatio * farHalfWidth, farClipPlaneDistance),
        new SFVec3f ( farHalfWidth, -aspectRatio * farHalfWidth, farClipPlaneDistance),
        new SFVec3f (-farHalfWidth, aspectRatio * -farHalfWidth, farClipPlaneDistance),
        new SFVec3f (-farHalfWidth, -aspectRatio * farHalfWidth, farClipPlaneDistance));

    if (trace) Browser.println ('[' + scriptName + '] ' + 'output ' +
        '<Coordinate DEF="FrustrumCoordinate" ' +
        ' point=' + point_changed.toString() + '"' + '>');
}
}
```

Examine Viewpoint

Examine NavigationInfo

Compute Extrusion frustum

Compute Coordinate points for outline

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN" "http://www.web3d.org/specifications/x3d-3.2.dtd">
3 <X3D profile='Immersive' version='3.2' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifi
4 <head>
5   <meta content='ViewFrustrumExample.x3d' name='title' />
6   <meta content='Display view frustrum associated with a given pair of Viewpoint, NavigationInfo nodes' name='description' />
7   <meta content='Don Brutzman' name='creator' />
8   <meta content='16 August 2008' name='translated' />
9   <meta content='17 August 2008' name='modified' />
10  <meta content='ViewFrustrumPrototype.x3d' name='reference' />
11  <meta content='ViewFrustrumComputation.png' name='drawing' />
12  <meta content='ViewFrustrumOverheadView.png' name='image' />
13  <meta content='ViewFrustrumObliqueView.png' name='image' />
14  <meta content='view culling frustrum' name='subject' />
15  <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumExample.x3d' name='identifier' />
16  <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator' />
17  <meta content='../license.html' name='license' />
18 </head>
19 <Scene>
20 <ExternProtoDeclare appinfo='Display view frustrum associated with a given pair of Viewpoint NavigationInfo nodes'
21   name='ViewFrustrum' url='ViewFrustrumPrototype.x3d' "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumPrototype
22   <field accessType='initializeOnly' name='ViewpointNode' type='SFNode' />
23   <field accessType='initializeOnly' name='NavigationInfoNode' type='SFNode' />
24   <field accessType='inputOutput' name='lineColor' type='SFColor' />
25   <field accessType='inputOutput' name='frustrumColor' type='SFColor' />
26   <field accessType='inputOutput' name='transparency' type='SFFloat' />
27   <field accessType='inputOutput' appinfo='assumed ratio height/width' name='aspectRatio' type='SFFloat' />
28   <field accessType='initializeOnly' name='trace' type='SFBool' />
29 </ExternProtoDeclare>
30 <!-- Example use -->
31 <ProtoInstance name='ViewFrustrum'>
32   <fieldValue name='ViewpointNode'>
33     <Viewpoint DEF='TestViewpoint' fieldOfView='0.78' />
34   </fieldValue>
35   <fieldValue name='NavigationInfoNode'>
36     <NavigationInfo DEF='TestNavigationInfo' avatarSize='1 1.6 0.75' visibilityLimit='15' />
37   </fieldValue>
38   <fieldValue name='lineColor' value='0.9 0.9 0.9' />
39   <fieldValue name='frustrumColor' value='0.8 0.8 0.8' />
40   <fieldValue name='transparency' value='0.75' />
41 </ProtoInstance>
42 <Viewpoint description='Above view' orientation='1 0 0 -1.57' position='0 40 0' />
43 <Viewpoint description='Frustrum viewpoint' />
44 <Viewpoint description='Behind frustrum viewpoint' position='0 0 15' />
45 <NavigationInfo type='EXAMINE' "FLY" "ANY" />
46 <!-- Visualization assists -->
47 <Inline DEF='GridXZ' url='../Savage/Tools/Authoring/GridXZ_20x20Fixed.x3d' "https://savage.nps.edu/Savage/Tools/Authoring/GridXZ_20x20Fixed.x3d"
48 <Transform scale='5 5 5'>

```

field definitions,  
no initializations

fieldValue initializations  
override default values

# Additional Prototype Examples

Numerous prototypes and examples are available in the Savage archive, especially

- <https://savage.nps.edu/Savage/Tools/Animation>  
Arbitrary Axis Cylinder Sensor, Color Sequencer, Double Click Touch Sensor, Flying Text, Hidden Viewpoint, Material Choice, Material Toggle, Push Button, Relative Proximity Sensor, Slider Float, Slider Integer, Time Delay Sensor, Viewpoint Sequencer, Waypoint Interpolator
- <https://savage.nps.edu/Savage/Tools/Authoring>  
Animated Viewpoint Recorder, Single Type Conversion, View Position Orientation

[back to Table of Contents](#)

# Chapter Summary

# Chapter Summary

## Concepts

- Motivation and Functional Summary

## Functional Descriptions and Examples

- ProtoDeclare, ProtoInterface, ProtoBody and field declarations
- IS / connect linking of field interfaces to internals
- ExternProtoDeclare and field signatures
- ProtoInstance, containerField, fieldValue initializations
- Advanced examples: design and re-use

# Suggested exercises

Add a given external prototype declaration and instance to improve an already-existing scene

Write three prototypes of increasing complexity:

- No ProtoInterface, no field definitions
- One or more field definitions, no Script
- Multiple field definitions, multiple IS/connect, Script

Design a multiple fan-in fan-out prototype by emulating an existing X3D node while adding new functionality

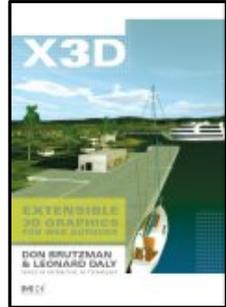
- Example: MaterialModulate

[back to Table of Contents](#)

# References

# References 1

*X3D: Extensible 3D Graphics for Web Authors*  
by Don Brutzman and Leonard Daly, Morgan  
Kaufmann Publishers, April 2007, 468 pages.



- Chapter 14, Creating Prototype Nodes
- <http://x3dGraphics.com>
- <http://x3dgraphics.com/examples/X3dForWebAuthors>

## X3D Resources

- <http://www.web3d.org/x3d/content/examples/X3dResources.html>

# References 2

## X3D-Edit Authoring Tool

- <https://savage.nps.edu/X3D-Edit>

## X3D Scene Authoring Hints

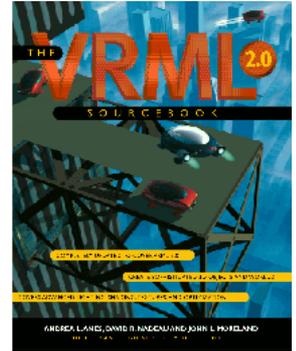
- <http://x3dgraphics.com/examples/X3dSceneAuthoringHints.html>  
(especially those for Inline and Prototypes)

## X3D Graphics Specification

- <http://www.web3d.org/x3d/specifications>
- Also available as help pages within X3D-Edit

# References 3

*VRML 2.0 Sourcebook* by Andrea L. Ames, David R. Nadeau, and John L. Moreland, John Wiley & Sons, 1996.



- <http://www.wiley.com/legacy/compbooks/vrml2sbk/cover/cover.htm>
- <http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook>
- Chapter 31 - Prototypes

# Contact

**Don Brutzman**

*brutzman@nps.edu*

*<http://faculty.nps.edu/brutzman>*

Code USW/Br, Naval Postgraduate School

Monterey California 93943-5000 USA

1.831.656.2149 voice

# CGEMS, SIGGRAPH, Eurographics

The Computer Graphics Educational Materials Source(CGEMS) site is designed for educators

- to provide a source of refereed high-quality content
- as a service to the Computer Graphics community
- freely available, directly prepared for classroom use
- <http://cgems.inesc.pt>

*X3D for Web Authors* recognized by CGEMS! 😊

- Book materials: X3D-Edit tool, examples, slidesets
- Received jury award for Best Submission 2008

CGEMS supported by SIGGRAPH, Eurographics



# Creative Commons open-source license

<http://creativecommons.org/licenses/by-nc-sa/3.0>



## Attribution-Noncommercial-Share Alike 3.0 Unported

**You are free:**

-  **to Share** — to copy, distribute and transmit the work
-  **to Remix** — to adapt the work

**Under the following conditions:**

-  **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
-  **Noncommercial.** You may not use this work for commercial purposes.
-  **Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

- ◆ For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.
- ◆ Any of the above conditions can be waived if you get permission from the copyright holder.
- ◆ Nothing in this license impairs or restricts the author's moral rights.

Disclaimer

Your fair dealing and other rights are in no way affected by the above.

# Open-source license for X3D-Edit software and X3D example scenes

<http://www.web3d.org/x3d/content/examples/license.html>

Copyright (c) 1995-2013 held by the author(s). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the names of the Naval Postgraduate School (NPS) Modeling Virtual Environments and Simulation (MOVES) Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# X3D Graphics for Web Authors

## Chapter 14

### Creating Prototype Nodes

*There are more things in heaven and earth, Horatio,  
than are dreamt of in your philosophy.*

William Shakespeare, Hamlet Act I Scene V



Here is the story of my high-school senior English project about building a **concordance** of Shakespeare's *Hamlet*. Building a concordance was a relatively new concept in 1974: first creating a full index of words in a document, then counting the occurrence of each word, and afterwards using that information to analyze the writing style of the author. At that time, this technique was being applied to try to determine whether the same author had written all of the plays attributed to Shakespeare.

In this case, my program was written in **Fortran** and run on an **IBM 1130**. It took several weeks to type in the entire play onto **punch cards** (with help from a pretty classmate). Typing mistakes usually meant retyping the entire card; this was before time sharing and personal accounts with disk space. Because the dataset was considered quite large, we were only able to test the concordance-creation program in small batches. Columbia High School's data processing department provided an empty **hard disk** (which was about as big as a garbage-can lid) to store the sorting data, then let us use the computer over the weekend... We started the job late Friday afternoon, reading in several thousand cards (i.e. lines of prose, one line per 80-character card) to disk and then starting the counting, sorting and cross-referencing routines. Output went to the line printer.

The job ran all weekend... At 7 am Monday morning I arrived early, excited and full of anticipation. Sure enough the lab was hot and the computer console was running steadily, with all of the memory-bit lights flashing on and off. There on the chain-drive line printer was page after page of concordance entries, word by word, listing word frequency and line references. That was the good news. However, checking the pages revealed that the program output had only produced words starting with letter "A" up to words somewhere in the middle of letter "C"... Gee whiz, there sure was a lot of alphabet left! We shut down the program and reopened the lab. Later that day in Shakespeare class, the teacher clapped and laughed, as did we all. This was an interesting lesson in the limits of brute-force programming, memory and computation.

# Contents

Chapter Overview and Concepts

Functional Descriptions and Examples

Chapter Summary

Suggested Exercises

References



# Chapter Overview



# Overview: Prototypes

## Concepts

- Motivation and Functional Summary

## Functional Descriptions and Examples

- ProtoDeclare, ProtoInterface, ProtoBody and field declarations
- IS / connect linking of field interfaces to internals
- ExternProtoDeclare and field signatures
- ProtoInstance, containerField, fieldValue initializations
- Advanced examples: design and re-use



[back to Table of Contents](#)

# Concepts



## Prototype motivation: extensibility

The X in X3D stands for Extensible: we have engineered the X3D standard for future growth

- Supporting innovation by individual authors, rather than waiting for future versions of the specification

Other extensibility mechanisms available:

- Inline node allows one scene to pull in other scenes, but without modification or customization
- Script node allows creation of arbitrary functionality that receives (and responds to) routed events

Prototypes create new full-fledged X3D nodes

- With field definitions, render capability, etc.



Editorial note. Regarding capitalization of the word “Extensible,” the Web3D Consortium follows the example of the Extensible Markup Language (XML) rather than some less-grammatical capitalization like eXtensible.

## Comparison with Inline node

Inline is easier to create and use

- Simply loads and inserts another X3D scene

Inline nodes are less flexible

- Cannot be customized when imported since there is no override mechanism for internal field values
- Events can be passed into, out of Inline scene at run time by using predefined IMPORT, EXPORT statements, for exposed internal nodes inside Inline

Prototypes are preferred if initialization values are needed, routing also works unambiguously



Inline nodes are easier to use, prototypes are a little harder to create but more powerful. Your mileage may vary (YMMV).

Often a good development technique is to test out an approach by simply creating, copying and pasting a scene subgraph a few times until the desired structure and field definitions are clear. Then encapsulating the functionality in a single ProtoDeclare can be simpler. Upon creating the prototype declaration, the example subgraphs are replaced by ProtoInstance nodes with appropriate fieldValue override values..

## Prototype functional summary

A Prototype creates a new full-fledged X3D node

- With field definitions, render capability, etc.

X3D prototypes provide a way for X3D authors to create new node definitions

- ProtoInstance allows repeated reuse of a new node
- Fields can be exposed and parameterized, allowing customization (unlike Inline which is fixed content)

Prototypes can be used within the scene where they are defined, or used externally

- ExternProtoDeclare gives reference to declaration



# Declaration versus instances

Prototype declarations can be thought of as defining a cookie-cutter for a new node

- ProtoDeclare constructs the definition
- Definition does not yet create an actual new node

Prototype instances are the actual copies of the new node which gets displayed

- Just as cookie cutter is used to create new cookies

ProtoDeclare  
is a  
template

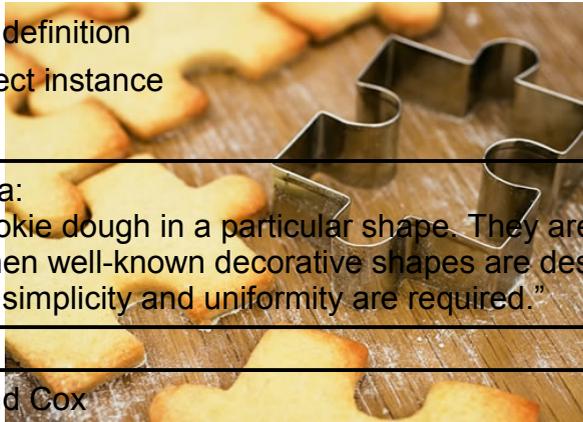
ProtoInstance  
copies actually  
exist and render

web|3D  
CONSORTIUM



In object-oriented parlance:

- ProtoDeclare corresponds to a class definition
- ProtoInstance corresponds to an object instance



From [Wikipedia](#), the free encyclopedia:

“A cookie cutter is a tool to cut out cookie dough in a particular shape. They are often used for seasonal occasions when well-known decorative shapes are desired, or for large batches of cookies where simplicity and uniformity are required.”

Image: Jigsaw Cookie Cutter, Cox and Cox

“Little ones will love helping out in the kitchen with this metal jigsaw piece cutter. Especially as they're allowed to play with their food! It provides endless fun for kids and is popular with adults, too. Imagine the effect of pieces running down the centre of a party table, or individual jigsaw piece biscuits being decorated with different children's names.”

[http://www.coxandcox.co.uk/index.php?main\\_page=product\\_info&cPath=9&products\\_id=51](http://www.coxandcox.co.uk/index.php?main_page=product_info&cPath=9&products_id=51)

## Summary of xml element structure

<b>ProtoDeclare</b> <ul style="list-style-type: none"><li>• ProtoInterface<ul style="list-style-type: none"><li>• field</li></ul></li><li>• ProtoBody<ul style="list-style-type: none"><li>• Initial node</li><li>• Additional nodes</li><li>• IS/connect links</li></ul></li></ul>	<b>Defines prototype</b> <ul style="list-style-type: none"><li>• Hold field definitions<ul style="list-style-type: none"><li>• Defines each field interface</li></ul></li><li>• Hold nodes, scene subgraph<ul style="list-style-type: none"><li>• First node defines type, use</li><li>• Initial siblings not rendered</li><li>• Link interfaces to internal fields</li></ul></li></ul>
<b>ExternProtoDeclare</b> <ul style="list-style-type: none"><li>• field</li></ul>	<b>Retrieve external declaration</b> <ul style="list-style-type: none"><li>• List of fields without values</li></ul>
<b>ProtoInstance</b> <ul style="list-style-type: none"><li>• fieldValue</li></ul>	<b>Actual copy of prototype node</b> <ul style="list-style-type: none"><li>• Override default interface values</li></ul>

## Potential power

Prototypes are a powerful technique for extending the capabilities of X3D

Few computing languages provide authors with the capability to extend the core vocabulary of the language itself

In one sense, an scene author defining a prototype for a new node in a scene can be thought to have similar power as the X3D specification team which defines new nodes for everyone to use in X3D



## Strong typing of nodes

Each prototype declaration must contain at least one node in the prototype body

- First node is primary, defining type for prototype
- ProtoInstances can only appear where that primary node might be allowed to appear
- If primary node contains children, together they must define a valid scene subgraph

Subsequent sibling nodes can follow first node

- But are not rendered, nor do they affect node type

Thus prototype instances remain strongly typed

- Any errors are discoverable before run time

This strong typing is important because it ensures that any addition of prototypes into a valid X3D scene remains a valid X3D scene.

This also prevents contradictory errors, such as a Prototype representing a modified Material node appearing someplace other than within a Shape node.

## Syntax alert: contrast .x3d .x3dv

Syntax for prototype definition and usage is significantly different when comparing the XML (.x3d) and ClassicVRML (.x3dv) encodings

Functional correspondence remains identical

- Declaration, field definitions, instance creation, etc.

Book compares both forms of syntax in detail



Because the X3D syntax is more explicit and detailed, it is usually easier to follow.

ClassicVRML and VRML97 syntax are identical for prototypes.

[back to Table of Contents](#)

# Functional Descriptions and Examples



# ProtoDeclare

A prototype declaration includes two constructs:  
prototype interface and prototype body

```
<ProtoDeclare name='MyNewBlueMaterial'>  
  <ProtoInterface>  
    <field name='concentration' accessType='inputOutput' type='SFInt32'  
      Value='0.75' appinfo='how blue is my new Material, range 0..1' />  
  </ProtoInterface>  
  <ProtoBody>  
    <!-- First node in body determines node type of prototype-->  
    <Material />  
    <!-- Subsequent nodes do not render, but must be valid X3D subgraph -->  
    <Script DEF='CalculateNewBlueValueFromConcentration' />  
  </ProtoBody>  
</ProtoDeclare>
```

Corresponding ClassicVRML construct: PROTO, followed by name, as shown in Table 14.2, pp. 386-387.

# Naming considerations 1

Good naming is important for prototypes, fields

- Helps authors understand their intent and then utilize them correctly
- Naming-convention guidelines found in X3D Scene Authoring Hints

Only one declaration is allowed for each individual prototype node

- Cannot have conflicting same-name definitions from ProtoDeclare and/or ExternProtoDeclare
- Name collisions (i.e. "overloading") not allowed



Scene Authoring Hints are provided in X3D-Edit Help system and are online at <http://www.web3d.org/x3d/content/examples/X3dSceneAuthoringHints.html#NamingConventions>

## Naming considerations 2

Good test of a prototype name (or field name) is to use it in a sentence, to see if it makes sense

- “a MaterialModulator node mimics a Material node and modulate fields as an animation effect”
- Awkward names are revealed by awkward sentences
- Descriptions are helpful when added as *appinfo*

Good names provide clarity when thinking about, modifying, and debugging a scene

Best name is when no one asks what it means!

- Alternatively, questions imply need to improve



Scene Authoring Hints are provided in X3D-Edit Help system and are online at <http://www.web3d.org/x3d/content/examples/X3dSceneAuthoringHints.html#NamingConventions>

*appinfo* is a descriptive attribute that authors can define for *field* and prototype declarations. It is defined similarly to XML Schema *appinfo*.

Acknowledgement: Jeff Weekley came up with our (ironic) metric about how to tell if a name works. Thanks Jeff!

## Naming conventions, excerpted

CamelCaseNaming: capitalize each word, never use abbreviations, strive for clarity, be brief but complete  
startWithLowerCaseLetter when defining field names (i.e. attributes) for Prototypes, Scripts

Ensure consistent capitalization throughout

Use the underscore character ("\_") to indicate subscripts on mathematical variables. Otherwise avoid use of underscores since they look like whitespace when part of a URL address

Avoid use of hyphens ("-") since these are erroneously turned into subtraction operators when converted into class or variable names



Scene Authoring Hints are provided in X3D-Edit Help system and are online at <http://www.web3d.org/x3d/content/examples/X3dSceneAuthoringHints.html#NamingConventions>

## ProtoInterface and field declarations

<ProtoInterface> is section of <ProtoDeclare> that holds <field> definitions

- Which are the interface for the prototype
- Zero or more <field> definitions allowed
- <ProtoInterface> omitted if no <field> definitions

Same as <field> definitions for Script node

- Defines *name*, *type*, *accessType*, and initial *value*
- SFNode, MFNode initializations are contained elements
- initializeOnly, inputOutput fields must have initial value
- inputOnly, outputOnly fields have no initial value



Corresponding ClassicVRML construct: [ square brackets around field definitions ] as shown in Table 14.2, pp. 386-387.

Field-Type Names	Description	Default Values
SFBool	Single-Field boolean value	false (XML syntax) or FALSE (ClassicVRML syntax)
MFBool	Multiple-Field boolean array	Empty list
SFColor	Single-Field color value, RGB	0 0 0
MFColor	Multiple-Field color array, RGB	Empty list
SFColorRGBA	Single-Field color value, red-green-blue alpha (opacity)	0 0 0 0
MFColorRGBA	Multiple-Field color array, red-green-blue alpha (opacity)	Empty list
SFInt32	Single-Field 32-bit integer value	0
MFInt32	Multiple-Field 32-bit integer array	Empty list
SFFloat	Single-Field single-precision floating-point value	0.0
MFFloat	Multiple-Field single-precision floating-point array	Empty list
SFDouble	Single-Field double-precision floating-point value	0.0
MFDouble	Multiple-Field double-precision array	Empty list
SFImage	Single-Field image value	0 0 0 Contains special pixel-encoding values, see Chapter 5 for details

Table 14.3, page 388, *X3D Field Types and Default Values*

MFIImage	Multiple-Field image value	Empty list
SFNode	Single-Field node	Empty node, NULL
MFNode	Multiple-Field node array of peers	Empty list
SFRotation	Single-Field rotation value using 3-tuple axis, radian-angle form	0 0 1 0
MFRotation	Multiple-Field rotation array	Empty list
SFString	Single-Field string value	Empty string, representable as two adjacent quotation marks
MFString	Multiple-Field string array	Empty list
SFTime	Single-Field time value	-1, sentinel indicating no time value.
MFTime	Multiple-Field time array	Empty list
SFVec2f/SFVec2d	Single-Field 2-float/2-double vector value	0 0
MFVec2f/MFVec2d	Multiple-Field 2-float/2-double vector array	Empty list
SFVec3f/SFVec3d	Single-Field vector value of 3-float/3-double values	0 0 0
MFVec3f/MFVec3d	Multiple-Field vector array of 3-float/3-double values	Empty list

Table 14.3, page 388, *X3D Field Types and Default Values*

# ProtoBody

First node in ProtoBody is required and critical, defining the node type

- This node is how a ProtoInstance will appear to scene graph

Additional nodes are allowed, but not rendered

- This is how prototypes provide extensibility while maintaining strong node typing
- X3D-Edit will provide warning about this, unless author inserts a comment beforehand

No object-oriented “inheritance” but...

- first node in body can be a nested ProtoInstance

Corresponding ClassicVRML construct: { squiggly brackets around node declarations } as shown in Table 14.2, pp. 386-387.

Nested prototypes are interesting but a little bit risky... they are well defined and unambiguous according to the specification, but in practice, X3D players have had trouble implementing them correctly and consistently. So *caveat emptor*, “your mileage may vary” if you use this construct.

## Simple example: UniversalMedia excerpt 1

The Universal Media Materials archive provides a number of example materials

- Available as prototypes, or cut + paste
- Built in, selectable within X3D-Edit Material editor
- No ProtoInterface/fields needed, just ProtoBody

```
<ProtoDeclare name='ArtDeco00'>
  <ProtoBody>
    <Material ambientIntensity='0.25'
      diffuseColor='0.282435 0.085159 0.134462'
      emissiveColor='0.0 0.0 0.0' shininess='0.127273'
      specularColor='0.276305 0.11431 0.139857' transparency='0.0' />
  </ProtoBody>
</ProtoDeclare>
```

## Simple example: UniversalMedia excerpt 2

### Alternatively, ExternProto retrieval:

```
<ExternProtoDeclare name='ArtDeco00'  
  url="ArtDecoPrototypesExcerpt.x3d#ArtDeco00"  
  "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-  
  Prototypes/ArtDecoPrototypesExcerpt.x3d#ArtDeco00"  
  "http://www.web3d.org/x3d/content/examples/Basic/UniversalMediaMaterials/  
  ArtDecoPrototypes.x3d#ArtDeco00"/>
```

### Invocation is identical in either case:

```
<Shape>  
  <Appearance>  
    <ProtoInstance containerField='material' name='ArtDeco00'/>  
  </Appearance>  
  <Sphere DEF='Ball' radius='0.5'/>  
</Shape>
```



*containerField* tells parent node the  
node type of the contained ProtoInstance.



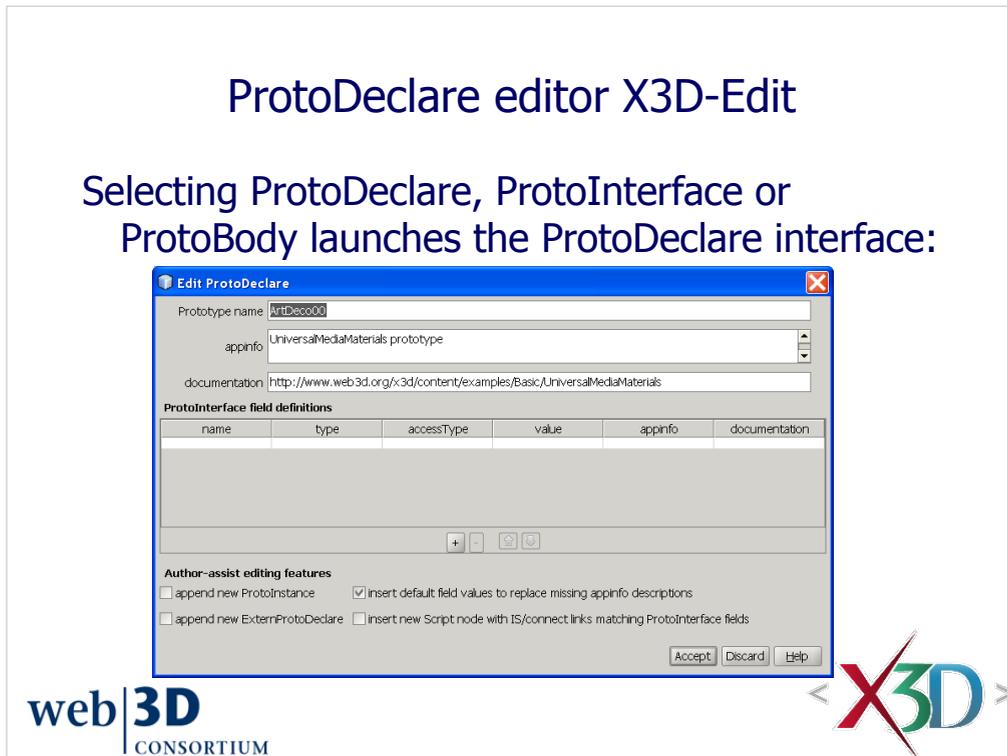
Note that *containerField*='material' is essential here to let the Shape know the node type of ArtDeco00. Otherwise the default *containerField*='children' is used by the browser, which is illegal inside a Shape node and would fail at run time.

```
ArtDecoPrototypesExcerpt.x3d - Editor
ArtDecoPrototypesExcerpt.x3d
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE X3D PUBLIC "-//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
3 <X3D profile="Immersive" version="3.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xsd:noNamespaceSchemaLocation="http://www.web3d.org/sp
4 <head>
5 <meta content="ArtDecoPrototypesExcerpt.x3d" name="title"/>
6 <meta content="Prototype declarations defining values for X3D/VRML materials, originally converted from SGI&apos;s Open Inventor material examp
7 <meta content="David Rousset" name="creator"/>
8 <meta content="James Harney, Don Brutzman WPS" name="translator"/>
9 <meta content="7 April 2002" name="created"/>
10 <meta content="18 November 2008" name="modified"/>
11 <meta content="http://vrlstuff.free.fr/materials/" name="reference"/>
12 <meta content="Universal Media Material Library" name="subject"/>
13 <meta content="http://www.web3d.org/x3d/content/examples/Basic/UniversalMediaMaterials/ArtDecoPrototypes.x3d" name="reference"/>
14 <meta content="http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ArtDecoPrototypesExcerpt.x3d" name="identifier"/>
15 <meta content="Vml97ToX3dNlist, http://ovrt.nist.gov/v2_x3d.html" name="generator"/>
16 <meta content="../license.html" name="license"/>
17 </head>
18 <scene>
19 <ProtoDeclare name="ArtDeco00">
20 <ProtoBody>
21 <Material ambientIntensity='0.25' diffuseColor='0.282435 0.085159 0.134462' emissiveColor='0.0 0.0 0.0' shininess='0.127273' specularColor=
22 </ProtoBody>
23 </ProtoDeclare>
24 <ProtoDeclare name="ArtDeco01">
25 <ProtoBody>
26 <Material ambientIntensity='0.254777' diffuseColor='0.685208 0.134679 0.332385' emissiveColor='0.0 0.0 0.0' shininess='0.071429' specularCo
27 </ProtoBody>
28 </ProtoDeclare>
29 <ProtoDeclare name="ArtDeco02">
30 <!-- computed conversion ambientIntensity=1.745282, normalized to 1.0 -->
31 <ProtoBody>
32 <Material ambientIntensity='1.0' diffuseColor='0.536861 0.0529 0.245479' emissiveColor='0.0 0.0 0.0' shininess='0.832432' specularColor='0.
33 </ProtoBody>
34 </ProtoDeclare>
35 <Anchor description="ArtDecoPrototypeExample" parameter="target=_blank" url="http://X3dGraphics.com/examples/X3dF
36 <Shape>
37 <Appearance>
38 <!-- replace Material node with a corresponding Prototype -->
39 <ProtoInstance containerField="material" name="ArtDeco00"/>
40 </Appearance>
41 <Text string="ArtDecoPrototypesExcerpt.x3d" "is a Materials Prototype declaration file." "" "For an example scene using these nodes," "cli
42 <FontStyle justify="MIDDLE" MIDDLE" size="0.8"/>
43 </Text>
44 </Shape>
45 </Anchor>
46 </Scene>
```

<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ArtDecoPrototypesExcerpt.x3d>

## ProtoDeclare editor X3D-Edit

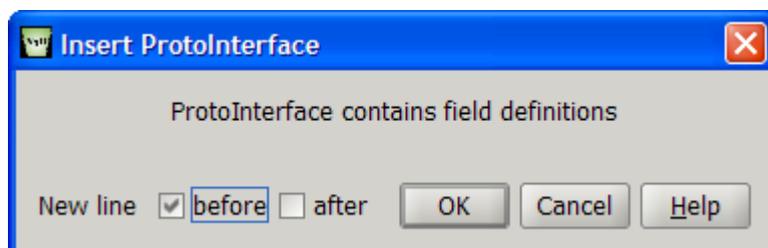
Selecting ProtoDeclare, ProtoInterface or ProtoBody launches the ProtoDeclare interface:



This example is very simple: there is no ProtoInterface and no field definitions.

ProtoInterface and ProtoBody are container elements only, with no attributes or independent functionality. Therefore there are no editor panes for these elements.

The ProtoInterface panel is minimalist, simply describing rules for use.



# Four prototype tooltips

		<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>P</b> ProtoBody	ProtoBody collects ProtoDeclare body nodes. <b>Warning:</b> only the first top-level node and its children are rendered, subsequent nodes (such as Scripts and ROUTEs) will be active but will not be drawn.	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>P</b> ProtoDeclare	ProtoDeclare is a Prototype declaration, defining a new node made up of other node(s). Hint: define field interfaces using the <field> tag, then scene nodes. Hint: initial scene node in a ProtoDeclare body determines this prototype's node type.	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
name	[name of the PROTO node being declared NMTOKEN #REQUIRED]	
appinfo	[appinfo type SFString CDATA #IMPLIED] Application information to provide simple description usable as a tooltip, similar to XML Schema appinfo tag.	
documentation	[documentation type SFString CDATA #IMPLIED] Documentation url for further information, similar to XML Schema documentation tag.	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>P</b> ProtoInstance	ProtoInstance creates a copy of a locally or externally defined PROTOTYPE node. Hint: override default initializations of field values using <fieldValue> tags. <b>Warning:</b> match PROTO node type to local context.	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
name	[name of the PROTO node being instanced NMTOKEN #REQUIRED]	
DEF	[DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model.	
USE	[USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. <b>Warning:</b> do NOT include DEF (or any other attribute values) when using a USE attribute!	
containerField	[containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.	
class	[class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>P</b> ProtoInterface	ProtoInterface collects ProtoDeclare field definitions.	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>

X3D Tooltips for ProtoBody, ProtoDeclare, ProtoInstance, ProtoInterface

<http://www.web3d.org/x3d/content/X3dTooltips.html#ProtoBody>

<http://www.web3d.org/x3d/content/X3dTooltips.html#ProtoDeclare>

<http://www.web3d.org/x3d/content/X3dTooltips.html#ProtoInstance>

<http://www.web3d.org/x3d/content/X3dTooltips.html#ProtoInterface>

		Top Resources Credits
<b>field</b>	A field element defines an interface attribute or node. Hint: first add Script, ProtoDeclare or ExternProtoDeclare before adding a field. Hint: put initializing SFNode/MFNode into contained content.	
name	[name: NMTOKEN #REQUIRED] Name of this field variable.	
accessType	[accessType: (inputOnly outputOnly initializeOnly inputOutput) #REQUIRED] Event-model semantics for field set/get capabilities. Hint for VRML 97: inputOnly=eventIn, outputOnly=eventOut, initializeOnly=field, inputOutput=exposedField. <b>Warning:</b> inputOutput=exposedField not allowed in VRML 97 Script nodes, use initializeOnly=field for backwards compatibility.	
type	[type: (select from types list) #REQUIRED] Base type of this field variable.	
value	[value: outputOnly CDATA #IMPLIED] Provide default initialization value for this field variable (may be later re-initialized by ProtoInstance fieldValue). Hint: SFNode/MFNode are initialized using contained content, instead of this value attribute. Hint: required for Script and ProtoDeclare. <b>Warning:</b> not allowed for ExternProtoDeclare. <b>Warning:</b> not allowed by inputOnly or outputOnly variables.	
appinfo	[appinfo type SFString CDATA #IMPLIED] Application information to provide simple description usable as a tooltip, similar to XML Schema appinfo tag.	
documentation	[documentation type SFString CDATA #IMPLIED] Documentation url for further information, similar to XML Schema documentation tag.	
		Top Resources Credits

X3D tooltips for *field*

<http://www.web3d.org/x3d/content/X3dTooltips.html#field>

## <IS> and <connect>

<IS><connect> definitions link field interfaces to internal nodes within the prototype body

These as direct links between outward-facing prototype interface fields and internal fields

- Any initialization or routed input value for the ProtoInterface field definition goes directly into matching internal IS/connect fields
- Any change to a connected internal field is routed out of the prototype, if *accessType*='outputOnly' or *accessType*='inputOutput'

Multiple connections are allowed for each node and for field, both for inputs and for outputs

Corresponding ClassicVRML construct: after field definition in prototype body, the keyword IS is appended, followed by name of corresponding field in proto interface, as shown in Table 14.4, pp. 389-391.

## <connect>

IS / connect constructs link field interfaces to internal nodes within the prototype declaration

- Each named field IS connected to a prototype field
- Only legal to use within ProtoBody declarations

Each <connect> definition provides connection between a given field within local parent node and a corresponding <field> definition in the ProtoInterface

- Each name must match field, interface exactly
- Identical (eponymous) names often best for clarity
- Must also match *type* and *accessType* exactly

Corresponding ClassicVRML construct: after field definition in prototype body, the keyword IS is appended, followed by name of corresponding field in proto interface

# <IS> and <connect> example

## Prototype interface fields linked to internal fields

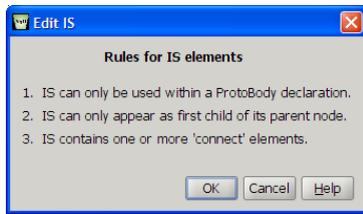
```
<ProtoDeclare appinfo='mimic a Material node and modulate fields as an animation effect'  
  name='MaterialModulator'>  
  <ProtoInterface>  
    <field accessType='inputOutput' name='enabled' type='SFBool' value='true'/>  
    <field accessType='inputOutput' name='diffuseColor' type='SFCOLOR' value='0.8 0.8 0.8'/>  
    <field accessType='inputOutput' name='emissiveColor' type='SFCOLOR' value='0 0 0'/>  
    <field accessType='inputOutput' name='specularColor' type='SFCOLOR' value='0 0 0'/>  
    <field accessType='inputOutput' name='transparency' type='SFFloat' value='0.0'/>  
    <field accessType='inputOutput' name='shininess' type='SFFloat' value='0.2'/>  
    <field accessType='inputOutput' name='ambientIntensity' type='SFFloat' value='0.2'/>  
  </ProtoInterface>  
  <ProtoBody>  
    <Material DEF='MaterialNode'>  
      <IS>  
        <connect nodeField='diffuseColor' protoField='diffuseColor'/>  
        <connect nodeField='emissiveColor' protoField='emissiveColor'/>  
        <connect nodeField='specularColor' protoField='specularColor'/>  
        <connect nodeField='transparency' protoField='transparency'/>  
        <connect nodeField='shininess' protoField='shininess'/>  
        <connect nodeField='ambientIntensity' protoField='ambientIntensity'/>  
      </IS>  
    </Material> <!-- etc. -->  
  </ProtoBody>  
</ProtoDeclare>
```

<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/MaterialModulator.x3d>

Note that you can <connect> multiple fields in a node to multiple protoFields, all at one time. Now we see why the <IS> element is used: to keep multiple <connect> definitions together.

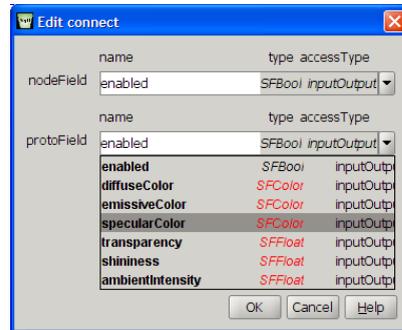
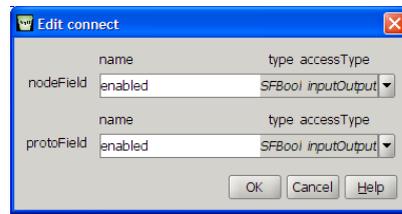
Question: hey, where is the *enabled* field hook up? Hmm, can't be hooked up to the Material since that node doesn't an *enabled* field of it's own. Must be connected somewhere else...

# IS / connect in X3D-Edit



<IS> editor is simple

<connect> editor prompts author to connect proper type and accessType between parent-node and prototype fields



		<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>IS</b>	<p>IS connects Prototype interface fields to node fields inside ProtoDeclare definitions. Add one or more connect tags to define each pair of Prototype field connections.</p> <p><b>Warning:</b> IS tag only allowed within ProtoDeclare body definitions.</p> <p><b>Hint:</b> IS tag precedes any Metadata tag, which precedes any other children tags.</p>	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
		<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>connect</b>	<p>connect tags define each Prototype field connection within ProtoDeclare definitions.</p> <p><b>Warning:</b> IS/connect tags are only allowed within ProtoDeclare body definitions.</p>	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>
<b>nodeField</b>	<p><b>[nodeField: NMTOKEN #REQUIRED]</b>  Name of field in this node connecting to parent ProtoDeclare field definition.  <b>Hint:</b> use multiple connect tags for multiple fan-in/fan-out.</p>	
<b>protoField</b>	<p><b>[protoField: NMTOKEN #REQUIRED]</b>  Name of parent ProtoDeclare field definition connecting to field in this node.  <b>Hint:</b> use multiple connect tags for multiple fan-in/fan-out.</p>	<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>

X3D Tooltips for IS, connect

<http://www.web3d.org/x3d/content/X3dTooltips.html#IS>

<http://www.web3d.org/x3d/content/X3dTooltips.html#connect>

# Connecting an embedded Script 1

A common design goal: create a Prototype that is modified version of specific node

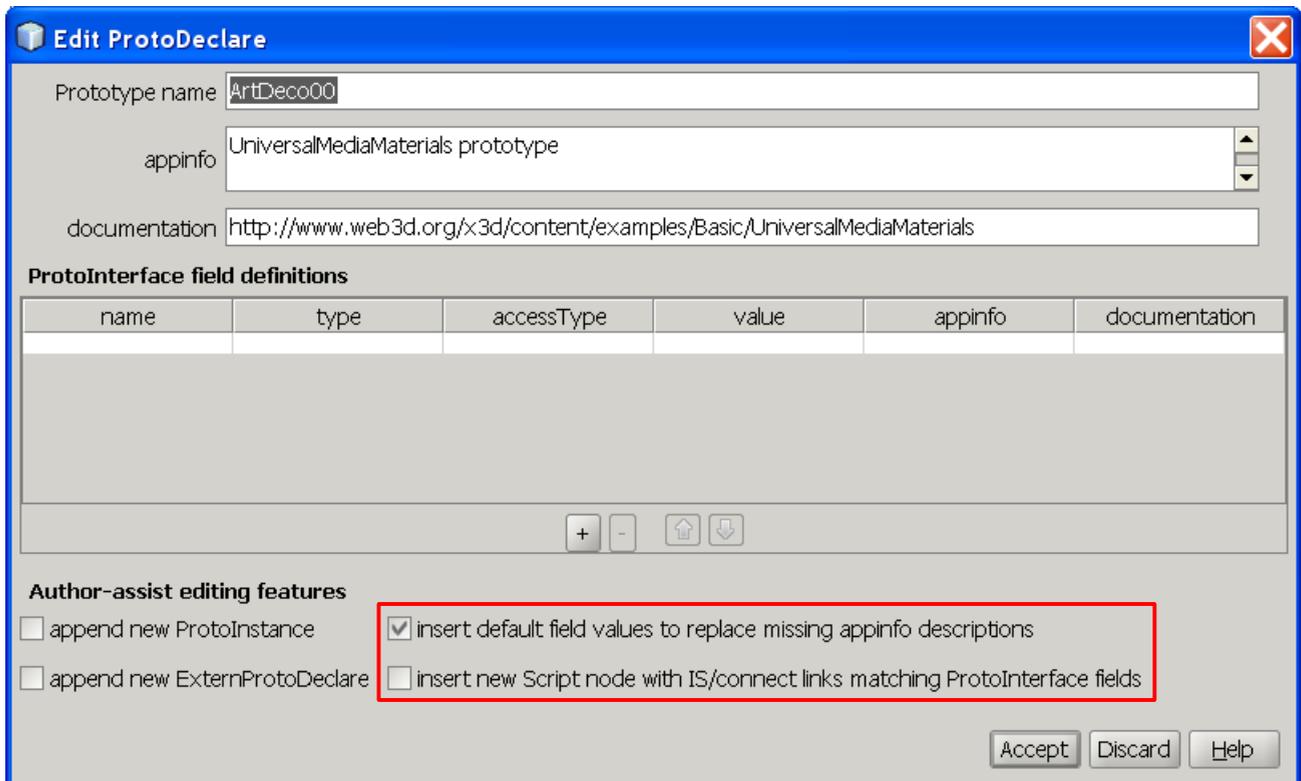
Example:

- Prototype name='NewMaterial'
- ProtoInterface holds definitions for all original fields plus possibly some additional fields
- ProtoBody initial node is essential: e.g. Material, fully linked by IS/connect definitions for each field
- Next (nonrendered) node is modifying Script, also holding IS/connect field definitions plus connection to Material (via ROUTE or DEF/USE in a field)

web|3D  
CONSORTIUM



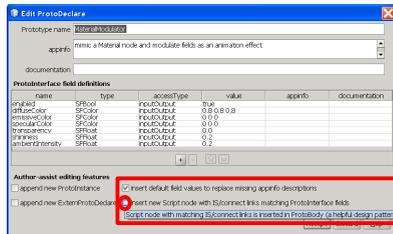
X3D-Edit feature: the ProtoDeclare editor offers an option to create a fully connected internal Script node by appropriately copying the prototype interface fields and then producing a Script containing corresponding field declarations and IS/connect definitions. When no *appinfo* is already provided, default values can be inserted.



# Connecting an embedded Script 2

X3D-Edit can insert Script if fields are defined

- May eventually add support for full design pattern



autogenerated X3D

```
<ProtoBody>
...
<Script DEF='MaterialModulatorScript'>
  <field accessType='inputOutput' name='enabled' type='SBool'/>
  <field accessType='inputOutput' name='diffuseColor' type='SColor'/>
  <field accessType='inputOutput' name='emissiveColor' type='SColor'/>
  <field accessType='inputOutput' name='specularColor' type='SColor'/>
  <field accessType='inputOutput' name='transparency' type='SFloat'/>
  <field accessType='inputOutput' name='shininess' type='SFloat'/>
  <field accessType='inputOutput' name='ambientIntensity' type='SFloat'/>
</IS>
  <connect nodeField='enabled' protoField='enabled'/>
  <connect nodeField='diffuseColor' protoField='diffuseColor'/>
  <connect nodeField='emissiveColor' protoField='emissiveColor'/>
  <connect nodeField='specularColor' protoField='specularColor'/>
  <connect nodeField='transparency' protoField='transparency'/>
  <connect nodeField='shininess' protoField='shininess'/>
  <connect nodeField='ambientIntensity' protoField='ambientIntensity'/>
</IS>
</Script>
</ProtoBody>
```



## ExternProtoDeclare

ExternProtoDeclare references an individual ProtoDeclare definition in an external scene

- Allows single “master” definition of a prototype, avoids versionitis from cut/paste redistributions
- Multiple prototype nodes require multiple ExternProtoDeclare statements

Includes <field> definitions matching interface signature of the original prototype

- Minus initial values, so that conflicts are avoided
- Allows X3D browser to “understand” new nodes and create proper scene graph when loading



Corresponding ClassicVRML construct: EXTERNPROTO, followed by name, as shown in Table 14.6, pp. 395-396.

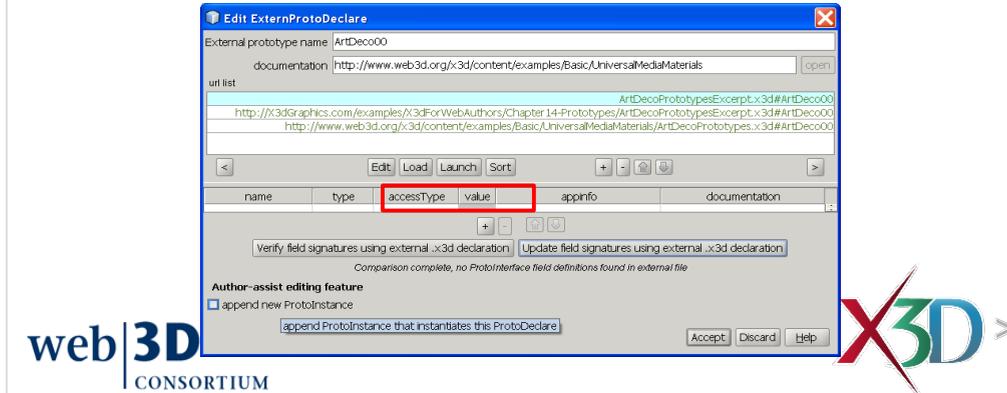
Some or all ExternProtoDeclare field definitions can be omitted if they are not initialized and not used by any of the corresponding ProtoInstance nodes.



## ExternProtoDeclare editor X3D-Edit

### ExternProtoDeclare editor for multiple url values

- Note #ProtoName appended to each filename
- Can edit, locally load, or launch each address
- Can sort url list (relative, .x3d before online, .wrl)



<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ArtDecoExamplesExcerpt.x3d>

Note that the *appinfo* field is typically a short description, suitable as a tool tip.

Note that the *documentation* field is typically a single url value linking to a help page.

A check button lets you confirm whether the ExternProtoDeclare definitions match the parent ProtoDeclare in a separate file. If there is a mismatch, the incorrect data fields are highlighted in red. The second button will then replace and fix any mismatches.

Loading the scene holding the referenced ProtoDeclare is sometimes convenient.

Author-assist editing feature allows you to append a corresponding new ProtoInstance that implements this ExternProtoDeclare.

TODO: add ... launch button for documentation url

## *appinfo, documentation* attributes

The *appinfo* and *documentation* attributes accompany ProtoDeclare, ExternProtoDeclare and field definitions

- *appinfo* holds a simple summary or tooltip
- *documentation* holds a url to further information

These match identical constructs in XML Schema

- Allowing tools to further support authoring, editing
- Allowing authors to properly document new nodes

These are important to use, and help long-term extensibility of your work and X3D itself



TODO under consideration: define X3D specification syntax for adding *appinfo* and *documentation* definitions to the ClassicVRML encoding.

## ProtoInstance

Finally you can make copies of your new node:  
create Prototype instances using ProtoInstance

- Must be preceded by either ProtoDeclare or ExternProtoDeclare with same name
- Otherwise a run-time error results for end user

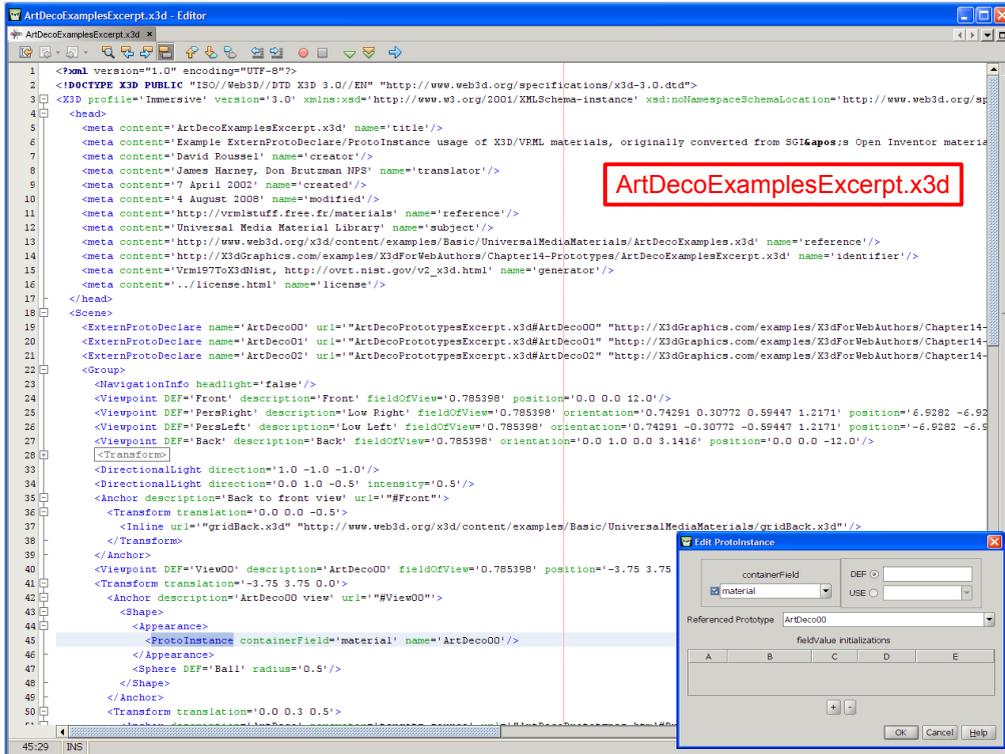
Nevertheless simple to invoke and instantiate:

```
<ProtoInstance name='ArtDeco00'/>
```

Can override default initialization values for fields

- This is how a prototype is customized upon creation
- `<fieldValue name='someField' value='someValue'/>`
- Can also initialize child nodes, if any

Corresponding ClassicVRML construct: no keyword, simply use of the prototype name when a node is expected, as shown in Table 14.7, page 398.



<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ArtDecoExamplesExcerpt.x3d>

Need ProtocalInstance editor snapshot (TODO, bug 1765, fails when no fieldValue given)

## *containerField* considerations

*containerField* is how the field name for a node is provided, relative to the node's parent

- Usually not needed since default matches most common case: *containerField* = 'children'
- ClassicVRML syntax is different, more verbose
- As ever, functionality is identical

<pre>&lt;!-- Rendered geometry follows prototype declaration --&gt; &lt;Shape&gt;   &lt;Sphere/&gt;   &lt;Appearance&gt;     &lt;ProtoInstance containerField='material'       name='MaterialModulator'&gt;       &lt;fieldValue name='enabled' value='true!'/&gt;       &lt;fieldValue name='diffuseColor' value='0.5 0.1 0.1'/&gt;     &lt;/ProtoInstance&gt;   &lt;/Appearance&gt; &lt;/Shape&gt;</pre>	<pre># Rendered geometry follows prototype declaration Shape {   geometry Sphere {   }   appearance Appearance {     material MaterialModulator {       enabled TRUE       diffuseColor 0.5 0.1 0.1     }   } }</pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

There have been a number of proposals to make ProtoInstance elements into “native node” elements, and to replace the containerField attribute with named elements, also called “wrapper tags.” Although these approaches have some interesting characteristics, they also have a significant number of drawbacks when applied to XML syntax.

The primary virtue of the ProtoInstance/containerField approach is that author-defined prototype instances can be validated by XML. By contrast, defining new XML elements that match the prototype names is visually appealing, but this approach quickly leads to nonvalidatable, erroneous content. So X3D doesn't do that.

# fieldValue initializations 1

fieldValue name must match; initialization values must match the type specified in declaration

- Otherwise a run-time error results for end user
- Take special care to check correctness, avoid errors

To initialize simple types: use *value* parameter

```
<ProtoInstance name='MaterialModulator'  
  containerField='material'  
  <fieldValue name='enabled' value='true'/>  
  <fieldValue name='diffuseColor' value='0.5 0.1 0.1'/>  
</ProtoInstance>
```



Re-using the same default initialization value is OK. Actually this is a common debugging technique when testing various combinations of field initialization values.

## fieldValue initializations 2

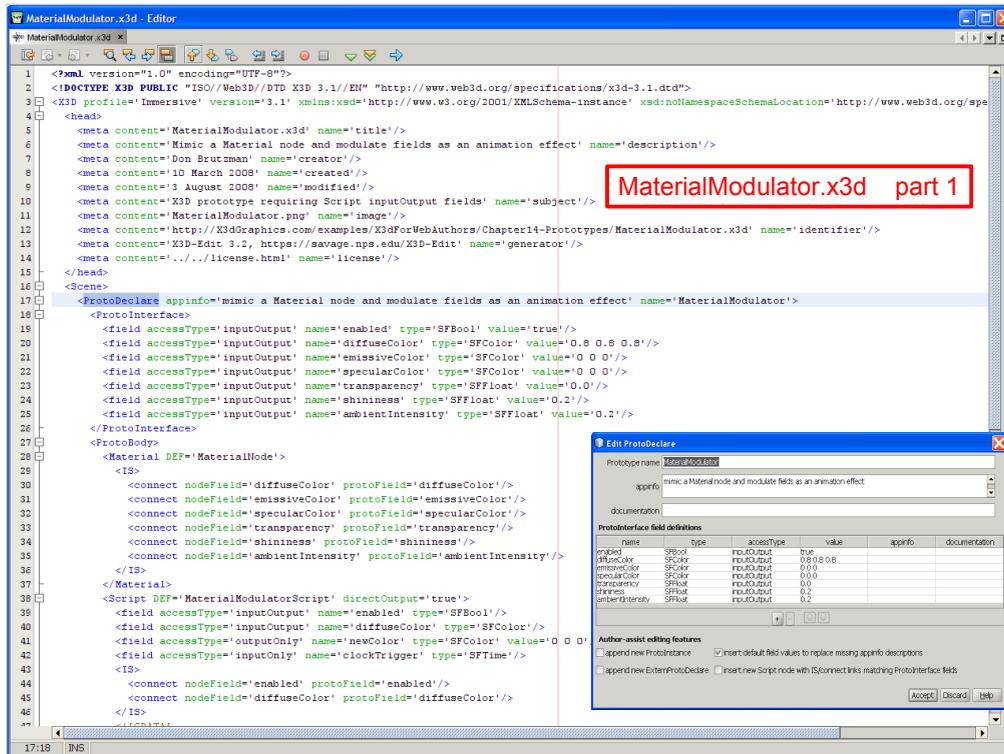
To initialize SFNode or MFNode types, use contained nodes within the fieldValue element:

```
<ProtoInstance name='SomethingNew'>
  <fieldValue name='newSFNodeField'>
    <!-- initialization node goes here -->
  </fieldValue>
</ProtoInstance>
```

As might be expected, fieldValue initializations are only allowed for fields with *accessType* of initializeOnly or inputOutput



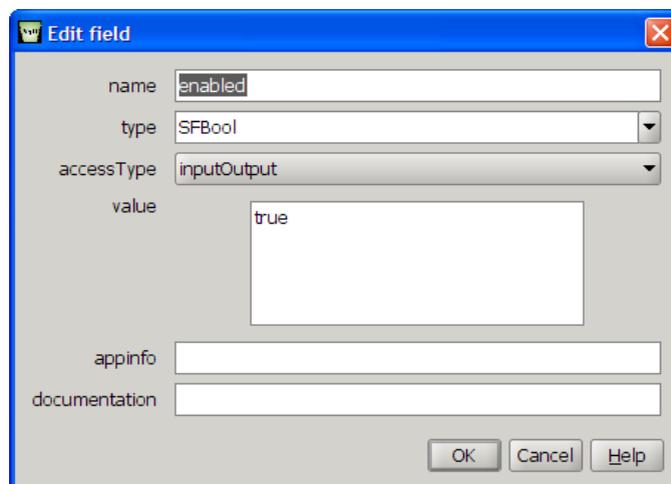
Re-using the same default initialization value is OK. Actually this is a common debugging technique when testing various combinations of field initialization values.

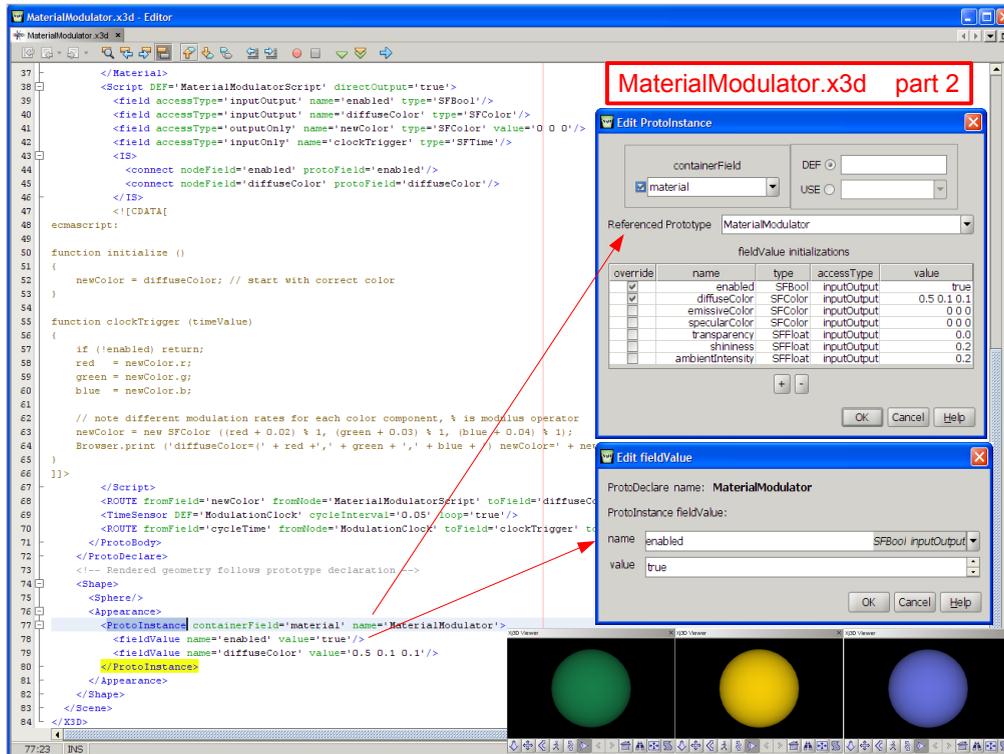


<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/MaterialModulator.x3d>

The ProtoDeclare editing panel provides a single interface to enter, view and change ProtoDeclare, ProtoInstance, and ProtoBody.

A separate panel for individual field editing is also provided:



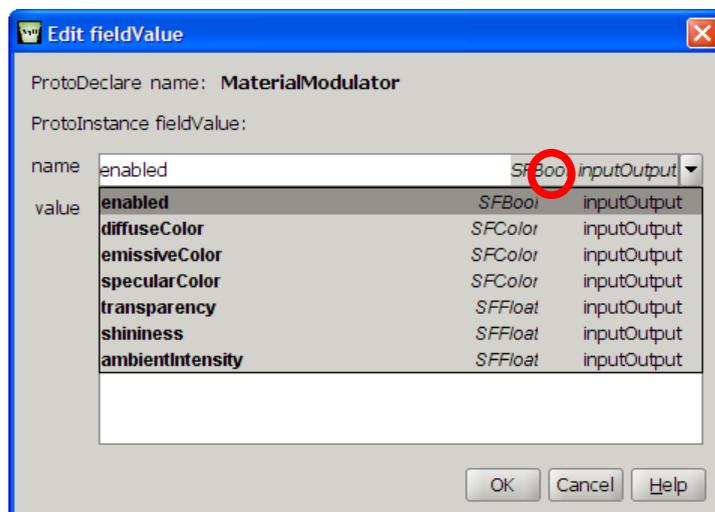


<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/MaterialModulator.x3d>

Note that the editing panel shows that only two <fieldValue> initializations are being overridden. The other <fieldValue> defaults are shown as a convenience.

The screen snapshot series in the lower right illustrate how the *diffuseColor* for the MaterialModulator nodes causes the Sphere appearance to change rapidly.

A separate panel for individual <fieldValue> editing is also provided. Note that it will list all available fields, allowing selection of the field of interest to be overridden. Here is the same <fieldValue> editing panel shown on the slide above, but with the author selecting the pull-down menu to choose the already-defined field of interest.



<b>P</b> ProtoInstance	ProtoInstance creates a copy of a locally or externally defined PROTOtype node. Hint: override default initializations of field values using <fieldValue> tags. <b>Warning:</b> match PROTO node type to local context.
name	[name of the PROTO node being instanced NMTOKEN #REQUIRED]
DEF	[DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model.
USE	[USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. <b>Warning:</b> do NOT include DEF (or any other attribute values) when using a USE attribute!
containerField	[containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.
class	[class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.
<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>	
<b>fieldValue</b>	A fieldValue element is used to re-initialize default field values in ProtoInstances. Field names must be already defined in ProtoDeclare or ExternProtoDeclare. Hint: put initializing SFNode/MFNode into fieldValue's contained content.
name	[name: NMTOKEN #REQUIRED] Name of this field (already defined in ProtoDeclare or ExternProtoDeclare).
value	[value: outputOnly CDATA #IMPLIED] Initial value for this field (overrides default initialization value in ProtoDeclare or ExternProtoDeclare). Hint: initialize SFNode/MFNode using contained content instead.
<a href="#">Top</a> <a href="#">Resources</a> <a href="#">Credits</a>	

## X3D Tooltips for ProtoInstance and *fieldValue*

<http://www.web3d.org/x3d/content/X3dTooltips.html#ProtoInstance>

<http://www.web3d.org/x3d/content/X3dTooltips.html#fieldValue>

[back to Table of Contents](#)

# Advanced Examples



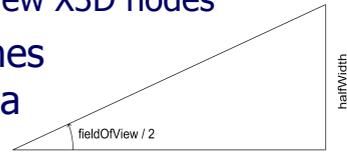
## Detailed example: ViewFrustrum

ViewFrustrum is a helpful visualization prototype

- Prototypes simplify creation of new X3D nodes

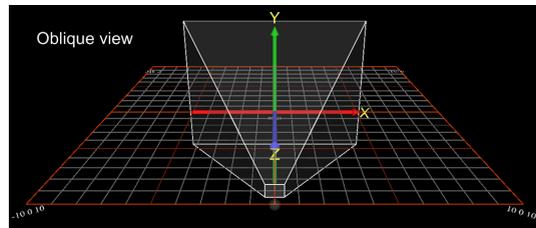
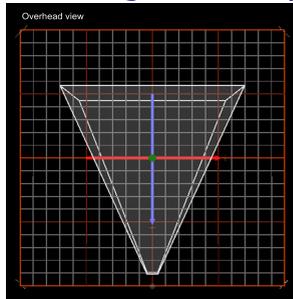
Shows near and far clipping planes that truncate the viewable area

- Depends on Viewpoint and NavigationInfo parameters



Near clipping plane distance = `avatarSize[0]`  
Far clipping plane distance = `visibilityLimit`

`nearHalfWidth = tan(fieldOfView / 2) * avatarSize[0];`  
`farHalfWidth = tan(fieldOfView / 2) * visibilityLimit;`

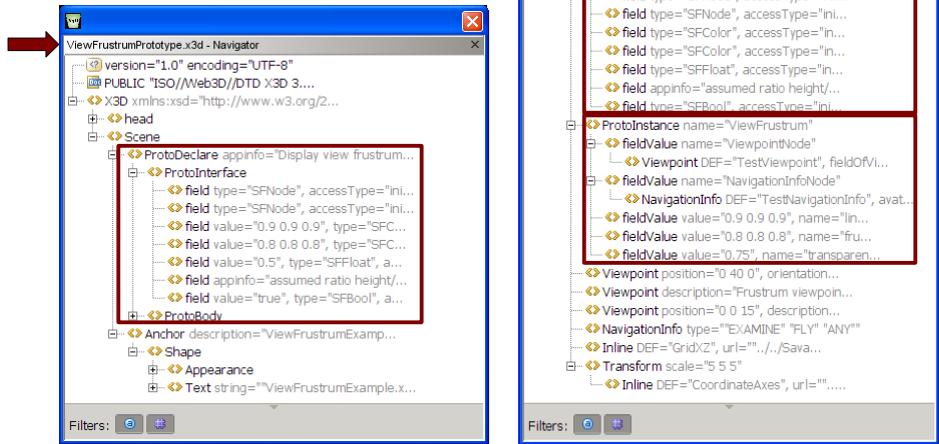


Viewpoint and NavigationInfo fields are covered in Chapter 4, Viewing and Navigation.

<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumExample.x3d>

# ViewFrustrum prototype, example

Good practice: make two separate files to simplify ExternProtoDeclare reuse



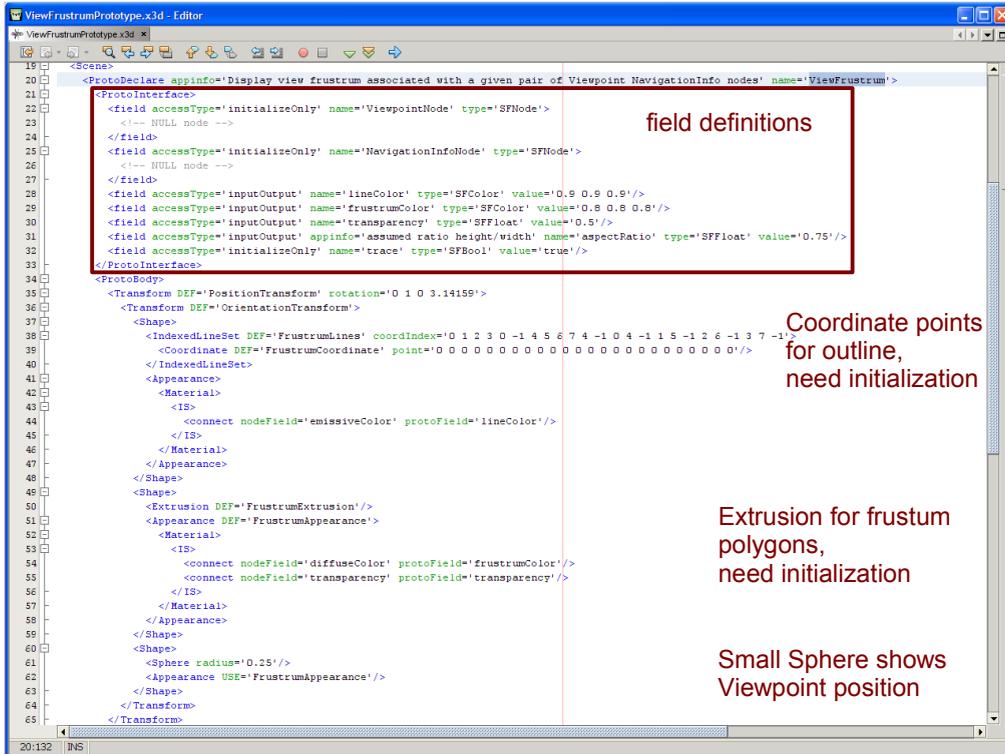
<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumPrototype.x3d>

<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumExample.x3d>

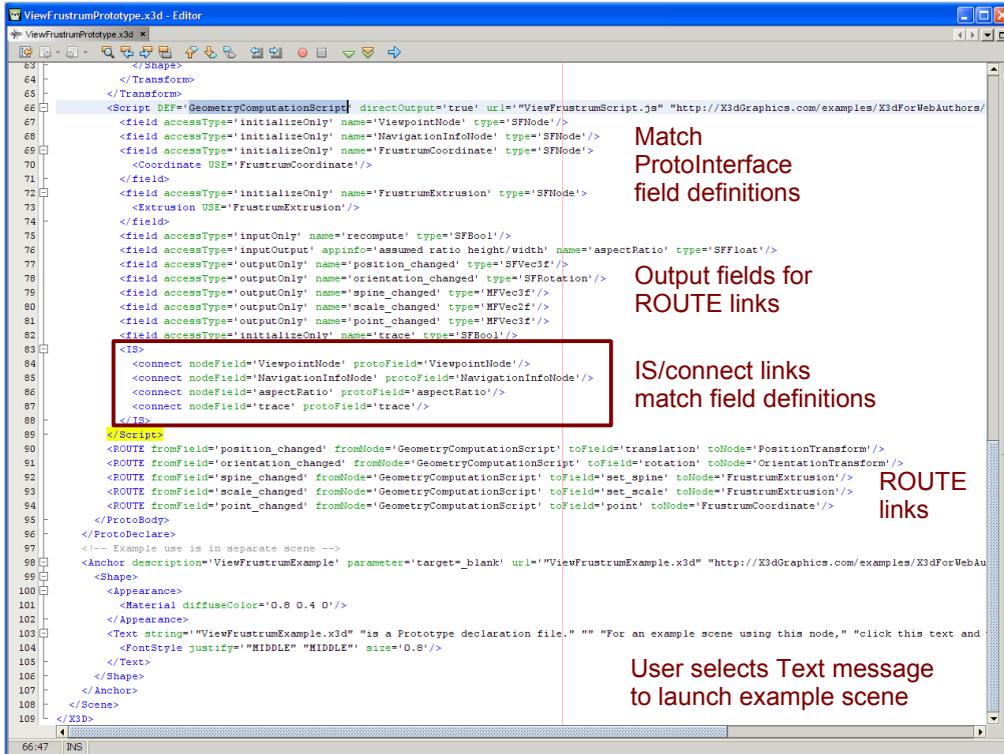
## Prototype features of interest

Highlighted ProtoDeclare, ExternProtoDeclare, ProtoInstance and Script show:

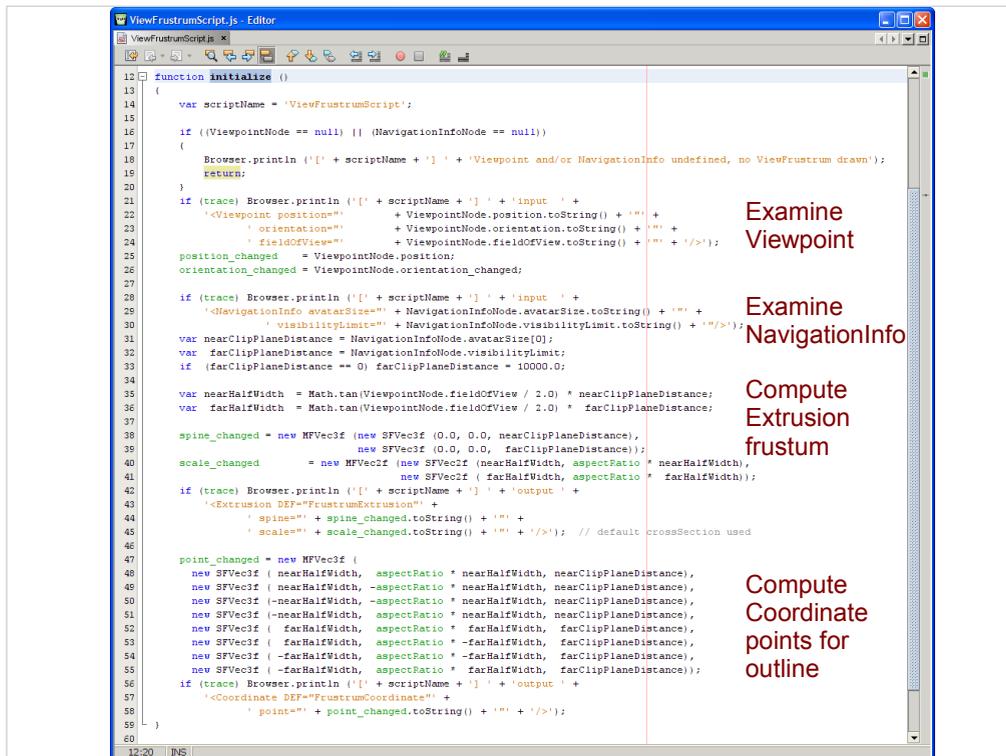
- Using `initialize()` method to setup geometry nodes
- Usage of `IS/connect` for direct node inspection
- Usage of event-passing via `ROUTE` when changing Extrusion, which doesn't support direct modification
- Matching *type* and *accessType*, `toString()` function
- External script code, accessing node fields
- Duplicate *url* addresses, local and remote
- `Browser.println` statements, silencable by *trace* field
- Internal `var` declarations, Javascript Math library



<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumPrototype.x3d>



<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumPrototype.x3d>



Editing the Script as a separate file provides Netbeans javascript syntax checking, code coloration, code completion, etc. This can catch a lot of errors.

### Script header:

```
// Description: Perform geometric computations for ViewFrustrum prototype
// Filename:      ViewFrustrumScript.js
// Author:       Don Brutzman
// Identifier:
// http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumScript.js
// Created:     16 August 2008
// Revised:    17 August 2008
// Reference:   ViewFrustrumPrototype.x3d
// Reference:   ViewFrustrumExample.x3d
// Drawing:    ViewFrustrumComputation.png
// License:    ../license.html
```

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE X3D PUBLIC "ISO/IEC15924:2004-08" "http://www.w3.org/2001/XMLSchema-instance" x3d:noNamespaceSchemaLocation="http://www.w3.org/2001/XMLSchema-instance" x3d:profile="immersive" version="3.2" xmlns:x3d="http://www.w3.org/2001/XMLSchema-instance" x3d:noNamespaceSchemaLocation="http://www.w3.org/2001/XMLSchema-instance" />
3
4 <head>
5   <meta content="ViewFrustrumExample.x3d" name="title"/>
6   <meta content="Display view frustrum associated with a given pair of Viewpoint, NavigationInfo nodes" name="description"/>
7   <meta content="Don Brutman" name="creator"/>
8   <meta content="16 August 2008" name="translated"/>
9   <meta content="17 August 2008" name="modified"/>
10  <meta content="ViewFrustrumPrototype.x3d" name="reference"/>
11  <meta content="ViewFrustrumComputation.png" name="drawing"/>
12  <meta content="ViewFrustrumOverheadView.png" name="image"/>
13  <meta content="ViewFrustrumObliqueView.png" name="image"/>
14  <meta content="view culling frustrum" name="subject"/>
15  <meta content="http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumExample.x3d" name="identifier"/>
16  <meta content="X3D-Edit, https://savage.nps.edu/X3D-Edit" name="generator"/>
17  <meta content="../license.html" name="license"/>
18 </head>
19 <!-- Scene -->
20 <Scene>
21   <!-- ExternProtoDeclare appinfo="Display view frustrum associated with a given pair of Viewpoint NavigationInfo nodes" -->
22   <ExternProtoDeclare appinfo="Display view frustrum associated with a given pair of Viewpoint NavigationInfo nodes"
23     name="ViewFrustrum" uri="ViewFrustrumPrototype.x3d" http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumPrototype.x3d">
24     <field accessType="initializeOnly" name="ViewpointNode" type="SFNode"/>
25     <field accessType="initializeOnly" name="NavigationInfoNode" type="SFNode"/>
26     <field accessType="inputOutput" name="lineColor" type="SFColor"/>
27     <field accessType="inputOutput" name="frustrumColor" type="SFColor"/>
28     <field accessType="inputOutput" name="transparency" type="SFFloat"/>
29     <field accessType="inputOutput" appinfo="assumed ratio height/width" name="aspectRatio" type="SFFloat"/>
30     <field accessType="initializeOnly" name="trace" type="SFBool"/>
31   </ExternProtoDeclare>
32   <!-- ProtoInstance -->
33   <ProtoInstance name="ViewFrustrum">
34     <fieldValue name="ViewpointNode">
35       <Viewpoint DEF="TestViewpoint" fieldOfView="0.78"/>
36     </fieldValue>
37     <fieldValue name="NavigationInfoNode">
38       <NavigationInfo DEF="TestNavigationInfo" avatarSize="1 1.6 0.75" visibilityLimit="15"/>
39     </fieldValue>
40     <fieldValue name="lineColor" value="0.9 0.9 0.9"/>
41     <fieldValue name="frustrumColor" value="0.8 0.8 0.8"/>
42     <fieldValue name="transparency" value="0.75"/>
43   </ProtoInstance>
44   <Viewpoint description="Above view" orientation="1 0 0 -1.57" position="0 0 0"/>
45   <Viewpoint description="Frustrum viewpoint"/>
46   <Viewpoint description="Behind frustrum viewpoint" position="0 0 15"/>
47   <NavigationInfo type="EXAMINE" "FLY" "ANY"/>
48   <!-- Visualization assists -->
49   <!-- GridXZ -->
50   <Inline DEF="GridXZ" uri="http://savage.nps.edu/Savage/Tools/Authoring/GridXZ_20x20Fixed.x3d" http://savage.nps.edu/Savage/Tools/Authoring/GridXZ_20x20Fixed.x3d">
51     <Transform scale="5 5 5">

```

field definitions,  
no initializations

fieldValue initializations  
override default values

<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustrumExample.x3d>

## Additional Prototype Examples

Numerous prototypes and examples are available in the Savage archive, especially

- <https://savage.nps.edu/Savage/Tools/Animation>  
Arbitrary Axis Cylinder Sensor, Color Sequencer, Double Click Touch Sensor, Flying Text, Hidden Viewpoint, Material Choice, Material Toggle, Push Button, Relative Proximity Sensor, Slider Float, Slider Integer, Time Delay Sensor, Viewpoint Sequencer, Waypoint Interpolator
- <https://savage.nps.edu/Savage/Tools/Authoring>  
Animated Viewpoint Recorder, Single Type Conversion, View Position Orientation



Each of these prototypes has both \_\_Prototypes.x3d and \_\_Examples.x3d scenes, showing ProtoDeclare definitions and separate ExternProtoDeclare invocations.

Looking at examples is very helpful for designing your own prototypes.

Each of these are maintained under version control and offered under an open-source license.

<https://savage.nps.edu/svn/nps/Savage>

[back to Table of Contents](#)

## Chapter Summary



# Chapter Summary

## Concepts

- Motivation and Functional Summary

## Functional Descriptions and Examples

- ProtoDeclare, ProtoInterface, ProtoBody and field declarations
- IS / connect linking of field interfaces to internals
- ExternProtoDeclare and field signatures
- ProtoInstance, containerField, fieldValue initializations
- Advanced examples: design and re-use



## Suggested exercises

Add a given external prototype declaration and instance to improve an already-existing scene

Write three prototypes of increasing complexity:

- No ProtoInterface, no field definitions
- One or more field definitions, no Script
- Multiple field definitions, multiple IS/connect, Script

Design a multiple fan-in fan-out prototype by emulating an existing X3D node while adding new functionality

- Example: MaterialModulate



[back to Table of Contents](#)

## References



# References 1

*X3D: Extensible 3D Graphics for Web Authors*  
by Don Brutzman and Leonard Daly, Morgan  
Kaufmann Publishers, April 2007, 468 pages.



- Chapter 14, Creating Prototype Nodes
- <http://x3dGraphics.com>
- <http://x3dgraphics.com/examples/X3dForWebAuthors>

## X3D Resources

- <http://www.web3d.org/x3d/content/examples/X3dResources.html>



# References 2

## X3D-Edit Authoring Tool

- <https://savage.nps.edu/X3D-Edit>

## X3D Scene Authoring Hints

- <http://x3dgraphics.com/examples/X3dSceneAuthoringHints.html>  
(especially those for Inline and Prototypes)

## X3D Graphics Specification

- <http://www.web3d.org/x3d/specifications>
- Also available as help pages within X3D-Edit



## Prototyping Excerpts from Scene Authoring Hints

### **Prototype Declarations**

- \* Follow X3D naming conventions for node and field definitions.
- \* Provide useful/safe default initialization values for each field, rather than depending on default field values internal to the ProtoBody.
- \* Include annotation tooltips for each field.
- \* Avoid copying ProtoDeclare definitions into scenes, instead copy ExternProtoDeclare/ProtoInstance definitions.
- \* Tooltips for ProtoDeclare, ProtoInterface and ProtoBody
- \* X3D specification

### **External Prototype Declarations**

- \* Do not wrap field definitions in a ProtoInterface element since that construct is illegal.
- \* For important prototypes, make a separate NewNodeExample.x3d scene that provides copyable/reusable ExternProtoDeclare/ProtoInstance definitions corresponding to each NewNodePrototype.x3d scene. This encourages authors to avoid copying ProtoDeclare definitions, so that a master version remains stable and improvable.
- \* Do not include initialization values in field definitions. They are illegal since the defaults in the original ProtoDeclare field declarations take precedence.
- \* Copy annotation tooltips from corresponding ProtoDeclare tooltips for each ExternProtoDeclare field.
- \* ExternProtoDeclare tooltips and X3D specification

### **Prototype Instances**

- \* Explicitly include initialization values, even if they match default values, to ensure proper operation. Sometimes a prototype can have different initialization values than expected, if it is modified elsewhere.
- \* Remember to include proper containerField attribute, identifying parent-node field name for this ProtoInstance. Default value: children. Example values: color, coord, geometry, fontStyle, proxy, sound, texture, textureTransform.
- \* First debug proper ProtoInstance operation in the scene defining the original ProtoDeclare, rather than using an ExternProtoDeclare. Why - to make sure they work first! Browser debugging can be more cryptic for externally defined prototypes and different versions may occur in various remote url addresses, making it difficult to determine precisely which ExternProtoDeclare is being referenced.
- \* ProtoInstance tooltips and X3D specification

## References 3

*VRML 2.0 Sourcebook* by Andrea L. Ames, David R. Nadeau, and John L. Moreland, John Wiley & Sons, 1996.



- <http://www.wiley.com/legacy/compbooks/vrml2sbk/cover/cover.htm>
- <http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook>
- Chapter 31 - Prototypes

# Contact

**Don Brutzman**

*[brutzman@nps.edu](mailto:brutzman@nps.edu)*

*<http://faculty.nps.edu/brutzman>*

Code USW/Br, Naval Postgraduate School  
Monterey California 93943-5000 USA  
1.831.656.2149 voice

web|**3D**  
CONSORTIUM



# CGEMS, SIGGRAPH, Eurographics

The Computer Graphics Educational Materials Source(CGEMS) site is designed for educators

- to provide a source of refereed high-quality content
- as a service to the Computer Graphics community
- freely available, directly prepared for classroom use
- <http://cgems.inesc.pt>

*X3D for Web Authors* recognized by CGEMS! ☺

- Book materials: X3D-Edit tool, examples, slidesets
- Received jury award for Best Submission 2008

CGEMS supported by SIGGRAPH, Eurographics



From the CGEMS home page:

- <http://cgems.inesc.pt>

Welcome to CGEMS - Computer Graphics Educational Materials Source. The CGEMS site is designed for educators to provide a source of refereed high-quality content as a service to the Computer Graphics community as a whole. Materials herein are freely available and directly prepared for your classroom.

List of all published modules:

- <http://cgems.inesc.pt/authors/ListModules.aspx>

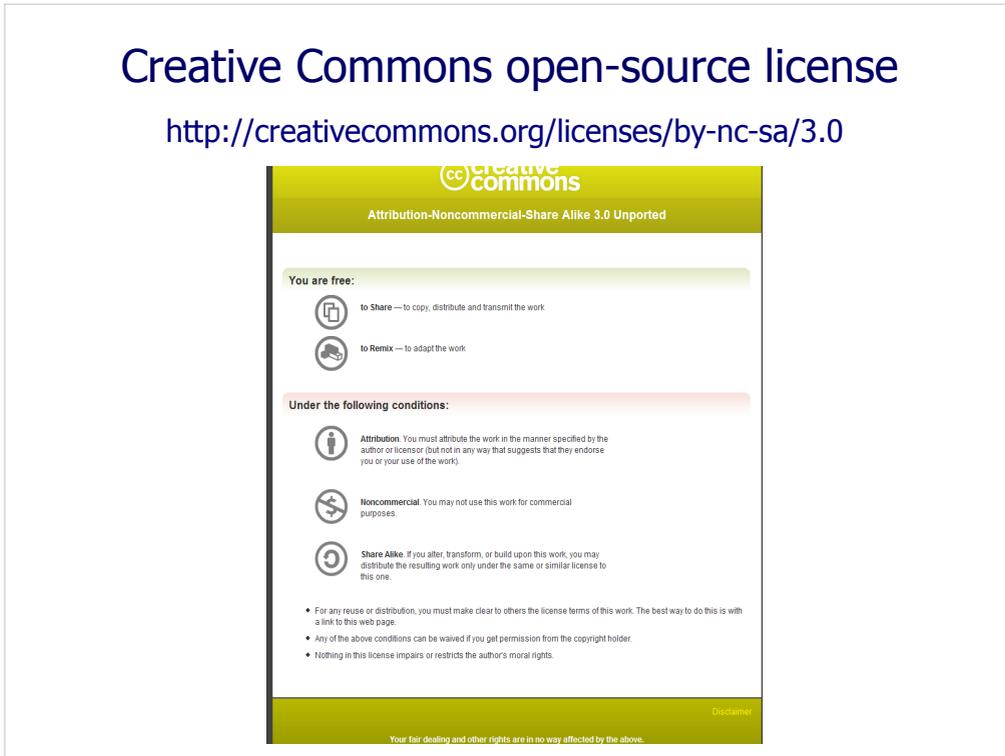


CGEMS Editorial Policy:

- <http://cgems.inesc.pt/EditorialPolicy.htm>

# Creative Commons open-source license

<http://creativecommons.org/licenses/by-nc-sa/3.0>



## Attribution-Noncommercial-Share Alike 3.0 Unported

You are free:

- \* to Share — to copy, distribute and transmit the work
- \* to Remix — to adapt the work

Under the following conditions:

\* Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Attribute this work: What does "Attribute this work" mean?

The page you came from contained embedded licensing metadata, including how the creator wishes to be attributed for re-use. You can use the HTML here to cite the work. Doing so will also include metadata on your page so that others can find the original work as well.

- \* Noncommercial. You may not use this work for commercial purposes.
- \* Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.
- \* For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.
- \* Any of the above conditions can be waived if you get permission from the copyright holder.
- \* Nothing in this license impairs or restricts the author's moral rights.

# Open-source license for X3D-Edit software and X3D example scenes

<http://www.web3d.org/x3d/content/examples/license.html>

Copyright (c) 1995-2013 held by the author(s). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the names of the Naval Postgraduate School (NPS) Modeling Virtual Environments and Simulation (MOVES) Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

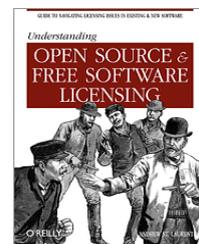
License available at

<http://www.web3d.org/x3d/content/examples/license.txt>

<http://www.web3d.org/x3d/content/examples/license.html>

Good references on open source:

Andrew M. St. Laurent, *Understanding Open Source and Free Software Licensing*, O'Reilly Publishing, Sebastopol California, August 2004. <http://oreilly.com/catalog/9780596005818/index.html>



Herz, J. C., Mark Lucas, John Scott, *Open Technology Development: Roadmap Plan*, Deputy Under Secretary of Defense for Advanced Systems and Concepts, Washington DC, April 2006. <http://handle.dtic.mil/100.2/ADA450769>

