X3D Graphics for Web Authors

Chapter 8

User Interactivity

Nobody knows the kind of trouble we're in.
Nobody seems to think it all might happen again.
Gram Parsons, "One Hundred Years from Now"

web3D CONSORTIUM

Contents

Chapter Overview

Overview: User Interactivity

User interactivity is initiated via sensor nodes, which capture user inputs and are hooked up to provide appropriate responses

- TouchSensor senses pointing device (mouse, etc.)
- PlaneSensor is a drag sensor that converts x-y pointer motion to move objects in a plane
- CylinderSensor and SphereSensor are drag sensors that convert x-y pointer motion to rotate objects
- KeySensor and StringSensor capture keyboard input

Interactivity sensors initiate animation chains

Related nodes

Chapter 4, Viewing and Navigation nodes

- Anchor: pointing device
  - Selects another Viewpoint or loads another scene
  - Show description when pointing device is over geometry
- Billboard rotates child geometry to face user
- Collision reports if viewer collides with geometry

Chapter 12, Environment Sensor and Sound

- LoadSensor reports when media asset is loaded
- ProximitySensor reports when user is in vicinity
- VisibilitySensor indicates when user's current camera view can see sensed geometry

Concepts

Importance of user interaction

Animated scenes are more interesting than static unchanging geometry

X3D interaction consists of sensing user actions and then prompting appropriate responses

Scenes that include behaviors which respond to user direction and control are more lively

Freedom of navigation and interaction contribute to user's sense of presence and immersion

Thus animation behaviors tend to be reactive and declarative, responding to the user

Sensors produce events

Sensors detect various kinds of user interaction and produce events to ROUTE within a scene

- Each sensor detects a certain kind of interaction, then produces one or more events

Authors decide how the events describing user interaction are interpreted and handled

- This approach allows great flexibility for authors

Using sensors in a scene

Three common design patterns (→ = ROUTE)

- Trigger (sensor) → Clock → Interpolator → Target node
- Sensor → Target node
- Sensor → Script (adaptor) → Target node

Pointing devices

Pointing device is primary tool for user interaction with geometry in a scene

- Mouse, Touchpad, touchscreen, or tracking stylus
- Arrow, Enter, other keys are allowed substitutes
- Trackball, data glove, game controller
- Tracking wand or other device in immersive 3D environments (such as a cave)
- Eye trackers and other advanced devices possible

X3D sensors designed for use with any generic pointing device, thus making scenes portable

Sensed geometry intersection, selection

Pointing devices communicate user's intended direction, movement, and selection (if any)

- Browsers and viewers usually superimpose 2D icon to indicate user's intended pointing direction
- 2D overlay icon may change to indicate selection

Sensors react to corresponding sibling and child geometry in the scene graph

- Pointing at other geometry means sensor activation no longer possible
- Usually one sensor must be deactivated before another can become active

Common field: *enabled*

enabled is an inputOutput boolean field that turns a sensor node on or off

- Thus allowing author to permit or disable flow of user-driven events which drive other responses
- Set *enabled*='false' to disrupt an event chain

Regardless of whether *enabled*='true' a sensor still needs a ROUTE connection from its output, or else no interaction response occurs

Common field: *isOver*

isOver is an outputOnly boolean field that reports when pointing at sensed geometry

- *isOver* true value sent when pointer is over shapes
- isOver false value sent when pointer icon is no longer over shapes
- If selection occurred, isOver false doesn't occur until after selection is released

Routing *isOver* values can enable animation

- Rapid sequencing on/off might remain a difficulty
- Take care that animation doesn't move viewpoint or geometry out from under the pointing device

Common field: *isActive*

isActive is an outputOnly boolean field that reports when sensor has received user input

- *isActive* true value sent when selection begins
- *isActive* false value sent when selection released
- Note that *isActive* true already occurs as a prerequisite when a sensor is initially enabled

Routing *isActive* values can enable, disable TimeSensor and other animation nodes

- Rapid sequencing on/off can be a difficulty, however
- BooleanFilter, BooleanToggle, BooleanTrigger also useful: Chapter 9 Event Utilities and Scripting

Common field: *description*

Each sensor's *description* field alerts users to the presence and intended purpose of each sensor

- Thus including a *description* is quite important, otherwise user is left to guess about responses
- Nevertheless many authors forget to include *description,* which inhibits interactivity

X3D Specification gives browsers flexibility about how *description* strings are displayed

- Overlay text, window-border text, perhaps audio

Dragging

Dragging means to select (activate) a sensed object, then to move the pointing device while the sensor is still activated

This user action causes continuous generation of output events while dragging motion occurs

- Click + drag + release = Select + hold + release

Several common fields

- *enabled, description, isActive, isOver, touchTime*

Three X3DDragSensorNode type sensors are

- CylinderSensor, PlaneSensor, SphereSensor

3D (6DOF) control using 2D devices

Selected objects are 3D, located in 3D space

- Which provides 6 degrees of freedom (DOF) for 3D object motion, e.g. (*x, y, z, roll, pitch, yaw*)

However most pointing devices only 2D control, since only movements are left-right, up-down

- Mouse, touchpad or touch screen, keyboard, etc.

Must map 2D output device to 3D/6DOF motions

- Each drag sensor thus defines how 2D motion is interpreted: surface of cylinder, plane, or sphere
- Hopefully authored in a manner intuitive to user

back to Table of Contents

X3D Nodes and Examples

TouchSensor node

TouchSensor affects adjacent geometry, provides basic pointing-device contact interaction

- Sends *isOver* true event when first pointed at
- Sends *isActive* true event when selected
- Sends *isActive* false event when deselected
- Sends *isOver* false event when no longer pointed at

Selection is deliberate action by user, for example

- Mouse, touchpad, touchscreen: left-click button
- Keyboard: <Enter> key
- 3D wand: selection button

Sensed geometry grouping 1

All geometry that is a peer (or children of peers) of the TouchSensor nodes can be sensed

Use a grouping node (Group, Transform, etc.) to isolate sensed geometry of interest

- Don't want to make entire scene selectable, otherwise interaction isn't very sophisticated

Can attach different sensors to self-explanatory geometry for different tasks. Examples:

- Light switch *isOver* gives name, click to change
- Billboarded Text or buttons for multiple controls

Sensed geometry grouping 2

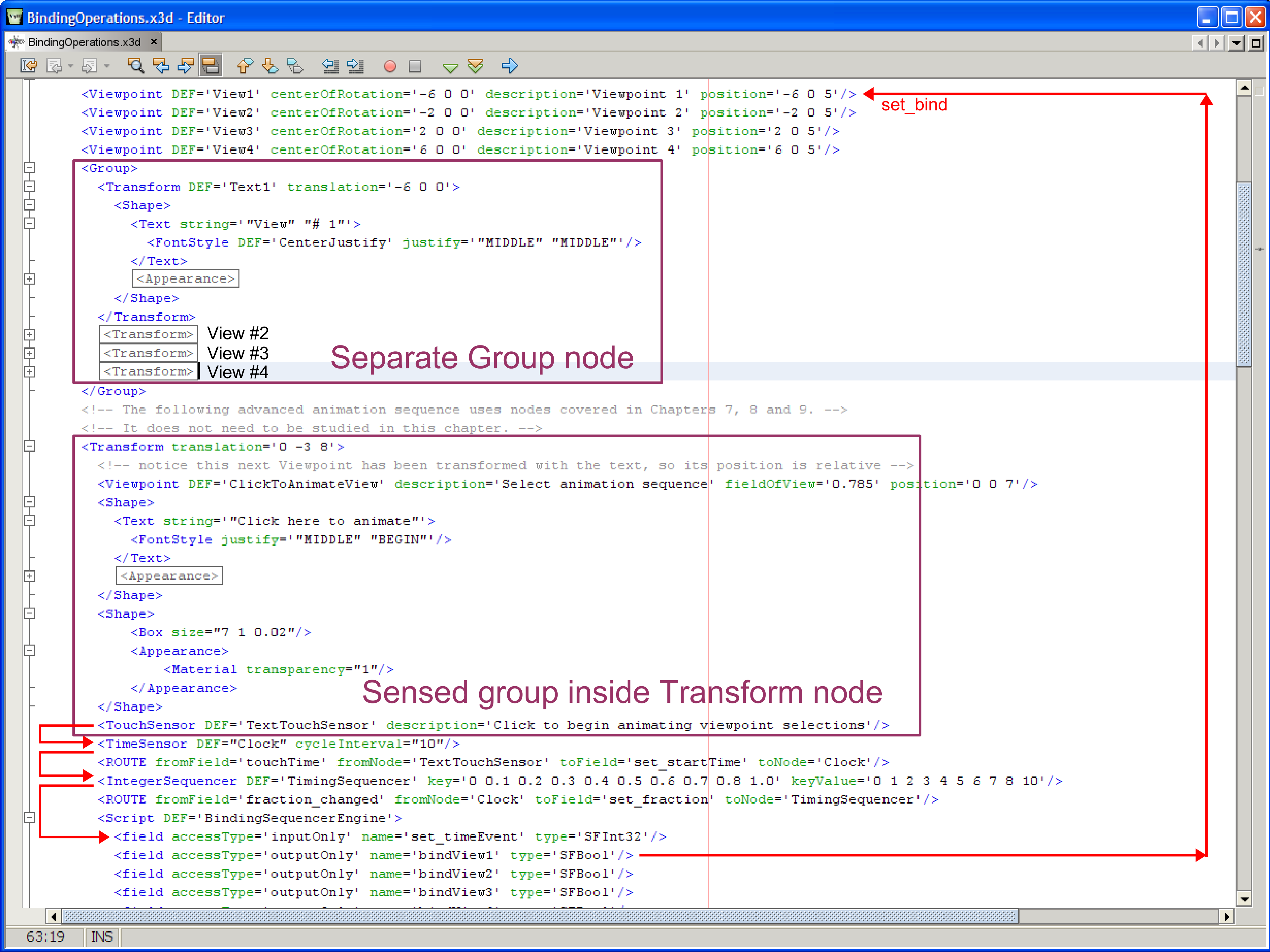
Separate sensed geometry from other shapes by using grouping nodes

Next slide shows example excerpt

- Chapter04-ViewingNavigation/BindOperations.x3d

Scene structure for this example

- Viewpoints consuming, producing events
- Display geometry, no sensor peer
- Selectable geometry, TouchSensor peer
- Regular animation design pattern: TimeSensor, Interpolator, target Script node, ROUTE connections


BindingOperations.x3d - Editor
BindingOperations.x3d
<Viewpoint DEF='View1' centerOfRotation='-6 0 0' description='Viewpoint 1' position='-6 0 5'/>
<Viewpoint DEF='View2' centerOfRotation='-2 0 0' description='Viewpoint 2' position='-2 0 5'/>
<Viewpoint DEF='View3' centerOfRotation='2 0 0' description='Viewpoint 3' position='2 0 5'/>
<Viewpoint DEF='View4' centerOfRotation='6 0 0' description='Viewpoint 4' position='6 0 5'/>
set_bind
<Group>
<Transform DEF='Text1' translation='-6 0 0'>
<Shape>
<Text string='"View" "# 1"'>
<FontStyle DEF='CenterJustify' justify='"MIDDLE" "MIDDLE"'/>
</Text>
<Appearance>
</Shape>
</Transform>
<Transform> View #2
<Transform> View #3
<Transform> View #4
Separate Group node
</Group>
<!-- The following advanced animation sequence uses nodes covered in Chapters 7, 8 and 9. -->
<!-- It does not need to be studied in this chapter. -->
<Transform translation='0 -3 8'>
<!-- notice this next Viewpoint has been transformed with the text, so its position is relative -->
<Viewpoint DEF='ClickToAnimateView' description='Select animation sequence' fieldOfView='0.785' position='0 0 7'/>
<Shape>
<Text string='"Click here to animate"'>
<FontStyle justify='"MIDDLE" "BEGIN"'/>
</Text>
<Appearance>
</Shape>
<Shape>
<Box size="7 1 0.02"/>
<Appearance>
<Material transparency="1"/>
</Appearance>
</Shape>
Sensed group inside Transform node
<TouchSensor DEF='TextTouchSensor' description='Click to begin animating viewpoint selections'/>
<TimeSensor DEF="Clock" cycleInterval="10"/>
<ROUTE fromField='touchTime' fromNode='TextTouchSensor' toField='set_startTime' toNode='Clock'/>
<IntegerSequencer DEF='TimingSequencer' key='0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 1.0' keyValue='0 1 2 3 4 5 6 7 8 10'/>
<ROUTE fromField='fraction_changed' fromNode='Clock' toField='set_fraction' toNode='TimingSequencer'/>
<Script DEF='BindingSequencerEngine'>
<field accessType='inputOnly' name='set_timeEvent' type='SFInt32'/>
<field accessType='outputOnly' name='bindView1' type='SFBool'/>
<field accessType='outputOnly' name='bindView2' type='SFBool'/>
<field accessType='outputOnly' name='bindView3' type='SFBool'/>
63:19
INS

Multiple TouchSensor nodes

Cannot sense just one part of grouped geometry

- Unless split out as separate groups of geometry, then Transform-ed to look like single shape to user

Can use multiple TouchSensor nodes, ROUTEs and event chains to accomplish multiple tasks

Can DEF, USE copies of single TouchSensor node, allowing multiple shapes to trigger same action

If multiple TouchSensor nodes at same level or above a given piece of geometry, nearest wins

- If tied at same distance, both activated at once

output event *touchTime*

touchTime sends an SFTime output event whenever sensed geometry is deselected

- Sent simultaneously with *isActive* false event

Three prerequisites must be met for *touchTime:*

1. Pointing device begins pointing at sensed geometry (generating *isOver* true event)
2. Pointing device is initially activated by user selection (generating *isActive* true event)
3. Pointing device is subsequently deactivated while still pointing at the sensed geometry (generating *isActive* false event)

output events *hitPoint_changed, hitNormal_changed, hitTexCoord_changed*

hitPoint_changed

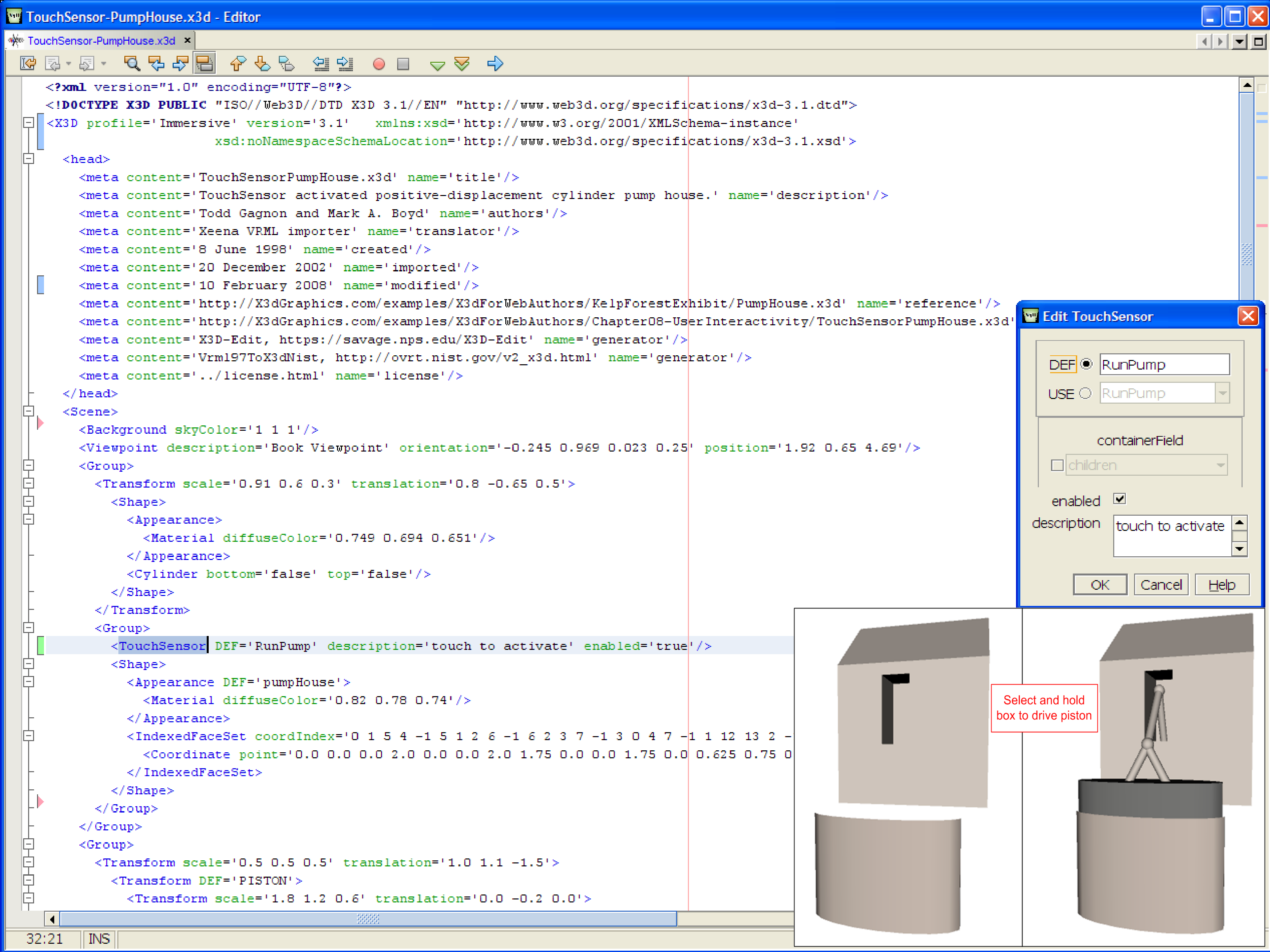
- sends output SFVec3f event providing 3D location coordinates of selection point, referenced to local coordinate system

hitNormal_changed

- sends output SFVec3f event providing normal vector of underlying geometry at selection point

hitTexCoord_changed

- sends output SFVec2f event providing 2D *(u, v)* coordinates of underlying texture at selection point



TouchSensor-PumpHouse.x3d - Editor
TouchSensor-PumpHouse.x3d
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.1.xsd'>
<head>
<meta content='TouchSensorPumpHouse.x3d' name='title'/>
<meta content='TouchSensor activated positive-displacement cylinder pump house.' name='description'/>
<meta content='Todd Gagnon and Mark A. Boyd' name='authors'/>
<meta content='Xeena VRML importer' name='translator'/>
<meta content='8 June 1998' name='created'/>
<meta content='20 December 2002' name='imported'/>
<meta content='10 February 2008' name='modified'/>
<meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/PumpHouse.x3d' name='reference'/>
<meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/TouchSensorPumpHouse.x3d'
<meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
<meta content='Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html' name='generator'/>
<meta content='../license.html' name='license'/>
</head>
<Scene>
<Background skyColor='1 1 1'/>
<Viewpoint description='Book Viewpoint' orientation='-0.245 0.969 0.023 0.25' position='1.92 0.65 4.69'/>
<Group>
<Transform scale='0.91 0.6 0.3' translation='0.8 -0.65 0.5'>
<Shape>
<Appearance>
<Material diffuseColor='0.749 0.694 0.651'/>
</Appearance>
<Cylinder bottom='false' top='false'/>
</Shape>
</Transform>
<Group>
<TouchSensor DEF='RunPump' description='touch to activate' enabled='true'/>
<Shape>
<Appearance DEF='pumpHouse'>
<Material diffuseColor='0.82 0.78 0.74'/>
</Appearance>
<IndexedFaceSet coordIndex='0 1 5 4 -1 5 1 2 6 -1 6 2 3 7 -1 3 0 4 7 -1 1 12 13 2 -
<Coordinate point='0.0 0.0 0.0 2.0 0.0 0.0 2.0 1.75 0.0 0.0 1.75 0.0 0.625 0.75 0
</IndexedFaceSet>
</Shape>
</Group>
</Group>
<Group>
<Transform scale='0.5 0.5 0.5' translation='1.0 1.1 -1.5'>
<Transform DEF='PISTON'>
<Transform scale='1.8 1.2 0.6' translation='0.0 -0.2 0.0'>
32:21
INS
Edit TouchSensor
DEF
RunPump
USE
RunPump
containerField
children
enabled
description
touch to activate
OK
Cancel
Help
Select and hold
box to drive piston

Example: opening doors

Interaction in 3D scenes doesn't always have to be literal. It is easier to click on a door to open it, rather than turning a door knob.

Next example compares TouchSensor selections

- Left door opens on initial selection   (click)
- Right door opens on later deselection (unclick)

Key difference: *isActive* is first true, then false

- To fix: routing events through a BooleanFilter and TimeTrigger can initiate TimeSensor appropriately
- These are Event Utility nodes, covered in Chapter 9

Doors.x3d - Editor

Doors.x3d

```
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/Doors.x3d' name='identifier'/>
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <Background skyColor='1 1 1'/>
    <NavigationInfo type='"WALK" "ANY"'/>
    <Viewpoint description='Initial default' position='0 1.4 10'/>
    <Viewpoint description='Book View' position='0 1.4 4.5'/>
    <Transform>   DoorRight
    <Transform DEF='DoorLeft' center='-.5 0 0' translation='-1 1.125 -.05'>
      <Group DEF='Door'>
        <Shape DEF='DoorShape'>
          <Appearance DEF='DoorApp'>
            <Material DEF='DoorMat' diffuseColor='0 .7 0'/>
            <ImageTexture DEF='DoorImage' url='"door_1.jpg" "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/door_
          </Appearance>
          <IndexedFaceSet DEF='DoorGeom' coordIndex='0 1 2 3 -1 4 7 6 5 -1 0 1 5 4 -1 1 5 6 2 -1 2 6 7 3 -1 3 7 6 4 -1'>
            <Coordinate point='-.5 -1.125 .05 .5 -1.125 .05 .5 1.125 .05 -.5 1.125 .05 -.5 -1.125 -.05 .5 -1.125 -.05 .5 1.125 -.05 -.5 1.125 -
            <TextureCoordinate point='0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1 1 0 .95 0 1 .95 .95 .95 1 0 .95 0 1 .95 .95 .95 1 0 .95 0 1 .95 .95 .95 1
          </IndexedFaceSet>
        </Shape>
      </Group>
      <TouchSensor DEF='TouchLeft' description='touch to activate' enabled='true'/>
      <BooleanFilter DEF='FilterLeft'/>
      <TimeTrigger DEF='TriggerLeft'/>
      <TimeSensor DEF='TimerLeft' cycleInterval='3' enabled='true' loop='false' startTime='0'/>
      <OrientationInterpolator DEF='MoverLeft' key='0 1' keyValue='0 1 0 0 0 1 0 -1'/>
      <ROUTE fromField='isActive' fromNode='TouchLeft' toField='set_boolean' toNode='FilterLeft'/>
      <ROUTE fromField='inputTrue' fromNode='FilterLeft' toField='set_boolean' toNode='TriggerLeft'/>
      <ROUTE fromField='triggerTime' fromNode='TriggerLeft' toField='startTime' toNode='TimerLeft'/>
      <ROUTE fromField='fraction_changed' fromNode='TimerLeft' toField='set_fraction' toNode='MoverLeft'/>
      <ROUTE fromField='value_changed' fromNode='MoverLeft' toField='rotation' toNode='DoorLeft'/>
    </Transform>
    <Transform DEF='DoorRight' center='-.5 0 0' translation='1 1.125 -.05'>
      <Group USE='Door'/>
      <TouchSensor DEF='TouchRight' description='touch to activate' enabled='true'/>
      <TimeSensor DEF='TimerRight' cycleInterval='3' enabled='true' loop='false' startTime='0'/>
      <OrientationInterpolator DEF='MoverRight' key='0 1' keyValue='0 1 0 0 0 1 0 -1'/>
      <ROUTE fromField='touchTime' fromNode='TouchRight' toField='startTime' toNode='TimerRight'/>
      <ROUTE fromField='fraction_changed' fromNode='TimerRight' toField='set_fraction' toNode='MoverRight'/>
      <ROUTE fromField='value_changed' fromNode='MoverRight' toField='rotation' toNode='DoorRight'/>
    </Transform>
  </Scene>
</X3D>
```

set_rotation

set_rotation

value_changed

value_changed

47:19 INS

TouchSensor	**TouchSensor tracks location & state of the pointing device, and detects when user points at geometry.** **Hint: Sensors are affected by peer nodes and children of peers.**
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. **Warning:** do NOT include DEF (or any other attribute values) when using a USE attribute!
description	**[description: accessType inputOutput, type SFString CDATA #IMPLIED]** Text description to be displayed for action of this node. **Hint:** use spaces, make descriptions clear and readable. **Hint:** many XML tools substitute XML character references automatically if needed (like & for & or " for ").
enabled	**[enabled: accessType inputOutput, type SFBool (true|false) "true"]** Enables/disables node operation.
isActive	**[isActive: accessType outputOnly, type SFBool (true|false) #FIXED ""]** Click or move the mouse (pointer) to generate isActive events. Event isActive=true is sent when primary mouse button is pressed. Event isActive=false is sent when primary mouse button is released.
isOver	**[isOver: accessType outputOnly, type SFBool (true|false) #FIXED ""]** is pointing device over sensor's geometry?
hitPoint_changed	**[hitPoint_changed: accessType outputOnly, type SFVec3f CDATA #FIXED ""]** Events containing 3D point on surface of underlying geometry, given in TouchSensor's local coordinate system.
hitNormal_changed	**[hitNormal_changed: accessType outputOnly, type SFVec3f CDATA #FIXED ""]** Events containing surface normal vector at the hitPoint.
hitTexCoord_changed	**[hitTexCoord_changed: accessType outputOnly, type SFVec2f CDATA #FIXED ""]** Events containing texture coordinates of surface at the hitPoint.
touchTime	**[touchTime: accessType outputOnly, type SFTime CDATA "0"]** Time event generated when sensor is touched by pointing device.
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.

PlaneSensor node

PlaneSensor converts x-y dragging motion by the pointing device into lateral translation in plane

- 2-tuple motion converted to 3-tuple SFVec3f
- Motion is parallel to local z=0 plane (screen plane)

Activated by peer geometry in scene graph

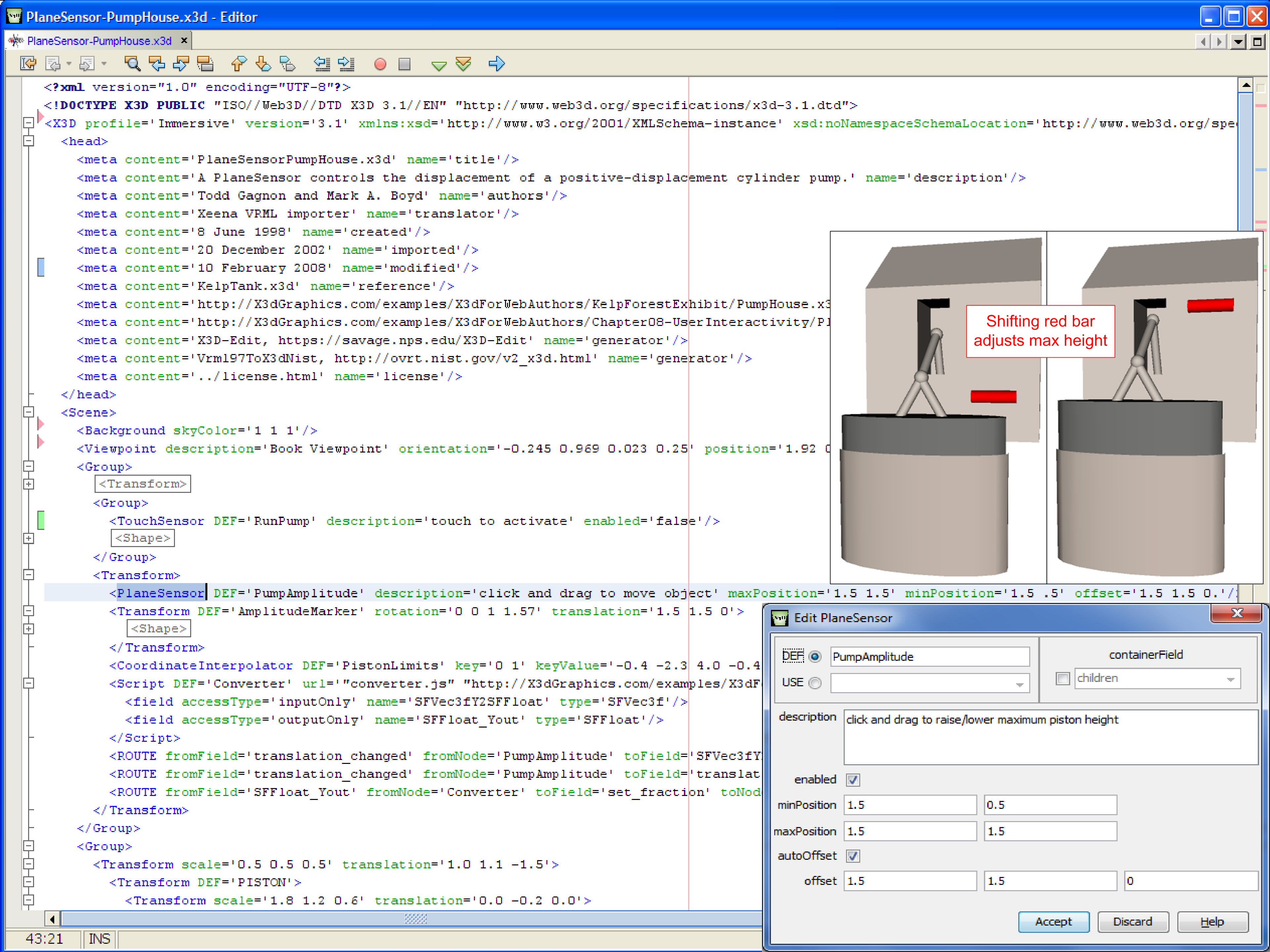
- Sensor itself is not rendered, unless background geometry or sensed shape itself has a planar side

Translation output values can follow a ROUTE connection to parent Transform *translation*

- Or connect to another SFVec3f field elsewhere

PlaneSensor fields, events

- Sends *isActive* true event when selected
- Sends *isActive* false event when deselected
- *minPosition, maxPosition* constrain X-Y translation to allowed planar region, defined as SFVec2f values
  - Example: *minPosition*='-2 -2' *maxPosition*='2 2'
- *offset* holds latest (or initial) SFVec3f position value
- *autoOffset*='true' remembers prior translation prior to resuming a new drag selection, otherwise *autoOffset*='false' jumps, restarts at initial position
- *translation_changed* and *trackPoint_changed* are the basic output events for sensor results


PlaneSensor-PumpHouse.x3d - Editor
PlaneSensor-PumpHouse.x3d
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/spe
<head>
<meta content='PlaneSensorPumpHouse.x3d' name='title'/>
<meta content='A PlaneSensor controls the displacement of a positive-displacement cylinder pump.' name='description'/>
<meta content='Todd Gagnon and Mark A. Boyd' name='authors'/>
<meta content='Xeena VRML importer' name='translator'/>
<meta content='8 June 1998' name='created'/>
<meta content='20 December 2002' name='imported'/>
<meta content='10 February 2008' name='modified'/>
<meta content='KelpTank.x3d' name='reference'/>
<meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/PumpHouse.x3
<meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/Pl
<meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
<meta content='Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html' name='generator'/>
<meta content='../license.html' name='license'/>
</head>
<Scene>
<Background skyColor='1 1 1'/>
<Viewpoint description='Book Viewpoint' orientation='-0.245 0.969 0.023 0.25' position='1.92 0
<Group>
<Transform>
<Group>
<TouchSensor DEF='RunPump' description='touch to activate' enabled='false'/>
<Shape>
</Group>
<Transform>
<PlaneSensor DEF='PumpAmplitude' description='click and drag to move object' maxPosition='1.5 1.5' minPosition='1.5 .5' offset='1.5 1.5 0.'/
<Transform DEF='AmplitudeMarker' rotation='0 0 1 1.57' translation='1.5 1.5 0'>
<Shape>
</Transform>
<CoordinateInterpolator DEF='PistonLimits' key='0 1' keyValue='-0.4 -2.3 4.0 -0.4
<Script DEF='Converter' url='"converter.js" "http://X3dGraphics.com/examples/X3dF
<field accessType='inputOnly' name='SFVec3fY2SFFloat' type='SFVec3f'/>
<field accessType='outputOnly' name='SFFloat_Yout' type='SFFloat'/>
</Script>
<ROUTE fromField='translation_changed' fromNode='PumpAmplitude' toField='SFVec3fY
<ROUTE fromField='translation_changed' fromNode='PumpAmplitude' toField='translat
<ROUTE fromField='SFFloat_Yout' fromNode='Converter' toField='set_fraction' toNod
</Transform>
</Group>
<Group>
<Transform scale='0.5 0.5 0.5' translation='1.0 1.1 -1.5'>
<Transform DEF='PISTON'>
<Transform scale='1.8 1.2 0.6' translation='0.0 -0.2 0.0'>
43:21
INS
Shifting red bar
adjusts max height
Edit PlaneSensor
DEF
PumpAmplitude
USE
containerField
children
description
click and drag to raise/lower maximum piston height
enabled
minPosition
1.5
0.5
maxPosition
1.5
1.5
autoOffset
offset
1.5
1.5
0
Accept
Discard
Help

TouchSensor	**TouchSensor tracks location & state of the pointing device, and detects when user points at geometry.** **Hint: Sensors are affected by peer nodes and children of peers.**
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. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute!
description	**[description: accessType inputOutput, type SFString CDATA #IMPLIED]** Text description to be displayed for action of this node. Hint: use spaces, make descriptions clear and readable. Hint: many XML tools substitute XML character references automatically if needed (like & for & or " for ").
enabled	**[enabled: accessType inputOutput, type SFBool (true|false) "true"]** Enables/disables node operation.
isActive	**[isActive: accessType outputOnly, type SFBool (true|false) #FIXED ""]** Click or move the mouse (pointer) to generate isActive events. Event isActive=true is sent when primary mouse button is pressed. Event isActive=false is sent when primary mouse button is released.
isOver	**[isOver: accessType outputOnly, type SFBool (true|false) #FIXED ""]** is pointing device over sensor's geometry?
hitPoint_changed	**[hitPoint_changed: accessType outputOnly, type SFVec3f CDATA #FIXED ""]** Events containing 3D point on surface of underlying geometry, given in TouchSensor's local coordinate system.
hitNormal_changed	**[hitNormal_changed: accessType outputOnly, type SFVec3f CDATA #FIXED ""]** Events containing surface normal vector at the hitPoint.
hitTexCoord_changed	**[hitTexCoord_changed: accessType outputOnly, type SFVec2f CDATA #FIXED ""]** Events containing texture coordinates of surface at the hitPoint.
touchTime	**[touchTime: accessType outputOnly, type SFTime CDATA "0"]** Time event generated when sensor is touched by pointing device.
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.

CylinderSensor node

CylinderSensor converts x-y dragging motion by the pointing device into rotation about an axis

- 2-tuple motion converted to 4-tuple SFRotation
- Rotation restricted to local coordinate frame y-axis

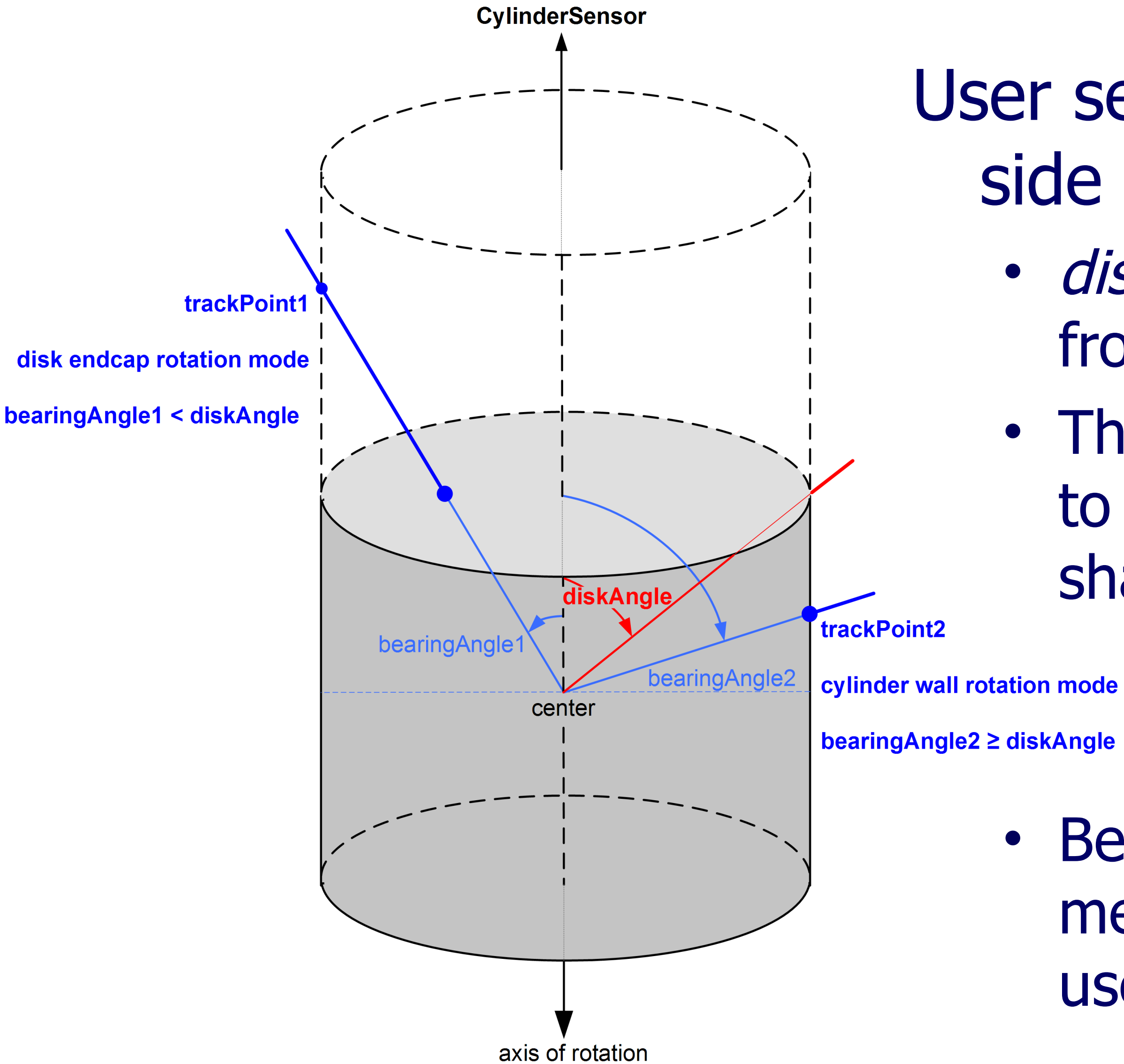
Activated by peer geometry in scene graph

- Sensor itself is not rendered, unless sensed shape is itself cylindrical

Rotation output values can follow a ROUTE connection to parent Transform *rotation*

- Or connect to another SFRotation field elsewhere

CylinderSensor *diskAngle* and select point determines tracking mode



User selects either end or side of drag cylinder

- *diskAngle* measures from axis to touch point
- Thus can adjust sensor to match cylindrical shape approximation
- Bearing angle is measured from axis to user's track point

CylinderSensor fields, events

- Sends *isActive* true event when selected
- Sends *isActive* false event when deselected
- *minAngle, maxAngle* constrain the allowed rotation
    - default values do need adjustment, always use radians
    - Example: *minAngle*='-3.14159' *maxAngle*='-3.14159'
- *offset* holds latest (or initial) rotation value
- *autoOffset*='true' remembers prior rotation prior to resuming a new drag selection, otherwise *autoOffset*='false' jumps to restart at initial rotation
- *rotation_changed* and *trackPoint_changed* are the basic output events for sensor results

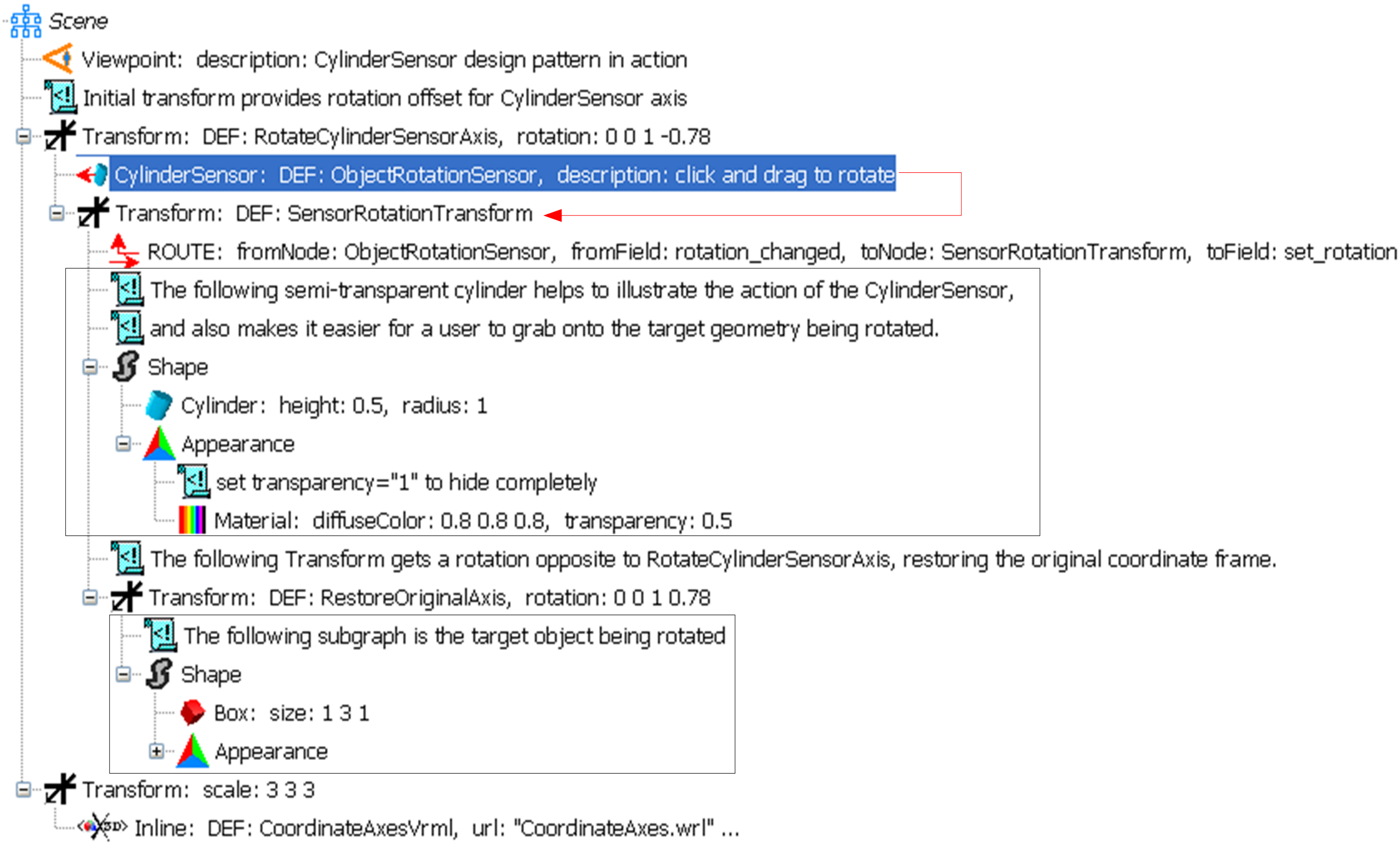
web3D CONSORTIUM

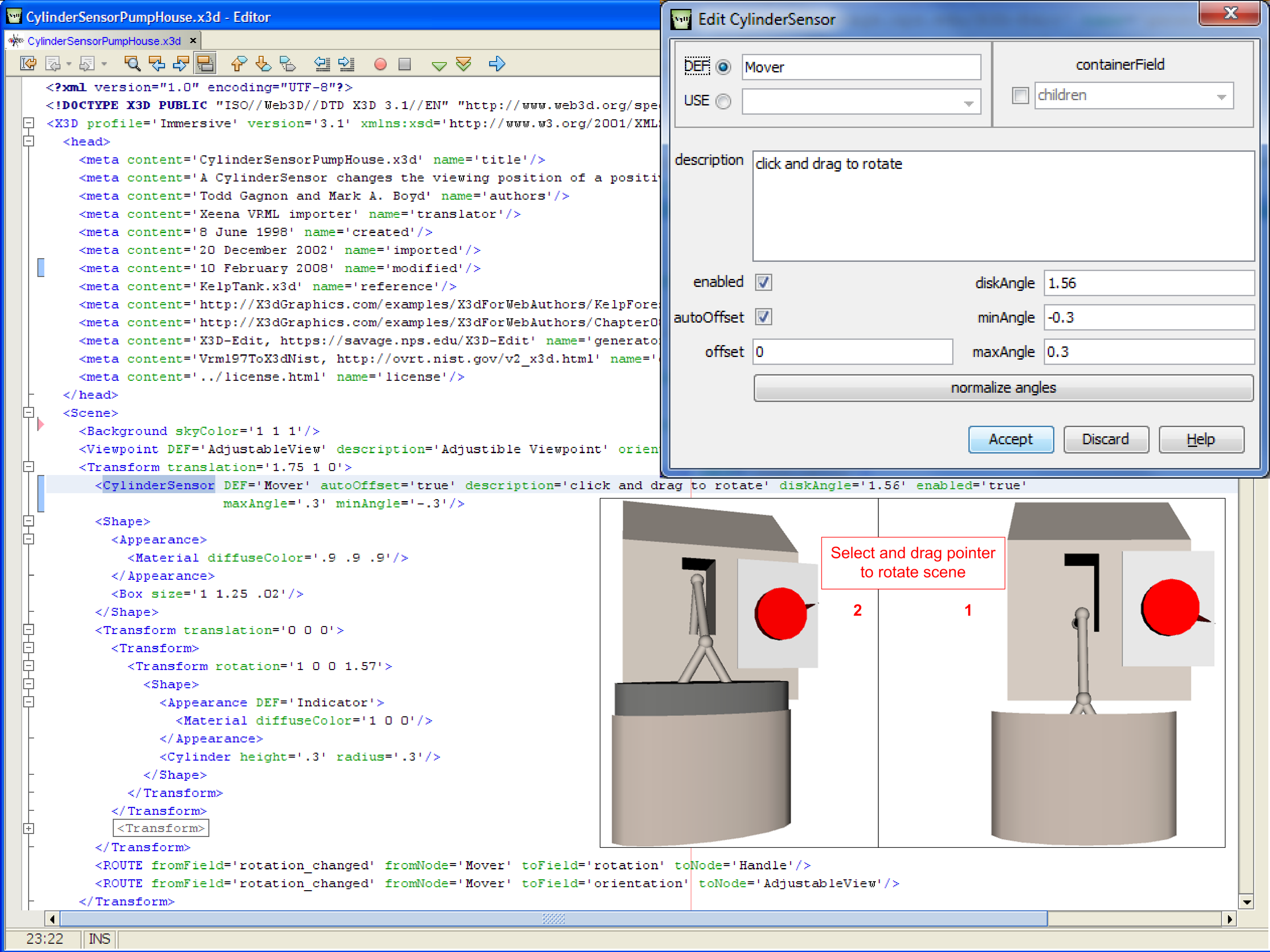
<X3D>

CylinderSensor off-axis rotation design pattern 1

- CylinderSensor rotates about the Y axis of local coordinate frame
  - No internal field provided for offsetting that axis
  - Making rotation axis different than peer sensed geometry can be tricky
- Following scene-graph design pattern shows how to rotate CylinderSensor about a different axis
  - First rotate to desired axis, CylinderSensor is child
  - Nest a second Transform rotation restoring original Y-axis, place sensed geometry here as child

CylinderSensor off-axis rotation design pattern 2




CylinderSensorPumpHouse.x3d - Editor
CylinderSensorPumpHouse.x3d
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/spe
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XML
<head>
<meta content='CylinderSensorPumpHouse.x3d' name='title'/>
<meta content='A CylinderSensor changes the viewing position of a positi
<meta content='Todd Gagnon and Mark A. Boyd' name='authors'/>
<meta content='Xeena VRML importer' name='translator'/>
<meta content='8 June 1998' name='created'/>
<meta content='20 December 2002' name='imported'/>
<meta content='10 February 2008' name='modified'/>
<meta content='KelpTank.x3d' name='reference'/>
<meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/KelpFore
<meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter0
<meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generato
<meta content='Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html' name='
<meta content='../license.html' name='license'/>
</head>
<Scene>
<Background skyColor='1 1 1'/>
<Viewpoint DEF='AdjustableView' description='Adjustible Viewpoint' orien
<Transform translation='1.75 1 0'>
<CylinderSensor DEF='Mover' autoOffset='true' description='click and drag to rotate' diskAngle='1.56' enabled='true'
maxAngle='.3' minAngle='-.3'/>
<Shape>
<Appearance>
<Material diffuseColor='.9 .9 .9'/>
</Appearance>
<Box size='1 1.25 .02'/>
</Shape>
<Transform translation='0 0 0'>
<Transform>
<Transform rotation='1 0 0 1.57'>
<Shape>
<Appearance DEF='Indicator'>
<Material diffuseColor='1 0 0'/>
</Appearance>
<Cylinder height='.3' radius='.3'/>
</Shape>
</Transform>
</Transform>
<Transform>
</Transform>
<ROUTE fromField='rotation_changed' fromNode='Mover' toField='rotation' toNode='Handle'/>
<ROUTE fromField='rotation_changed' fromNode='Mover' toField='orientation' toNode='AdjustableView'/>
</Transform>
23:22
INS
Edit CylinderSensor
DEF
Mover
USE
containerField
children
description
click and drag to rotate
enabled
autoOffset
offset
0
diskAngle
1.56
minAngle
-0.3
maxAngle
0.3
normalize angles
Accept
Discard
Help
Select and drag pointer
to rotate scene
2
1

CylinderSensor	**CylinderSensor converts pointer motion (for example, a mouse or wand) into rotation values using an invisible cylinder aligned with local Y-axis.** **Hint: Sensors are affected by peer nodes and children of peers.** **Hint: add transparent geometry to see the effect of the sensor.** **Hint: initial relative bearing of pointer drag determines whether cylinder sides or end-cap disks are used for manipulation.**
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. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute!
description	**[description: accessType inputOutput, type SFString CDATA #IMPLIED]** Text description to be displayed for action of this node. Hint: use spaces, make descriptions clear and readable. Hint: many XML tools substitute XML character references automatically if needed (like & for & or " for ").
enabled	**[enabled: accessType inputOutput, type SFBool (true|false) "true"]** Enables/disables node operation.
minAngle	**[minAngle: accessType inputOutput, type SFFloat CDATA "0"]** clamps rotation_changed events within range of min/max values Hint: if minAngle > maxAngle, rotation is not clamped.
maxAngle	**[maxAngle: accessType inputOutput, type SFFloat CDATA "0"]** clamps rotation_changed events within range of min/max values Hint: if minAngle > maxAngle, rotation is not clamped.
diskAngle	**[diskAngle: accessType inputOutput, type SFFloat CDATA "0.262" (15 degrees)]** Help decide rotation behavior from initial relative bearing of pointer drag: acute angle whether cylinder sides or end-cap disks of virtual-geometry sensor are used for manipulation. Hint: diskAngle 0 forces disk-like behavior, diskAngle 1.57 (90 degrees) forces cylinder-like behavior.
autoOffset	**[autoOffset: accessType inputOutput, type SFBool (true|false) "true"]** determines whether previous offset values are remembered/accumulated.
offset	**[offset: accessType inputOutput, type SFFloat CDATA "0"]** Sends event and remembers last value sensed.
isActive	**[isActive: accessType outputOnly, type SFBool (true|false) #FIXED ""]** isActive true/false events are sent when triggering the sensor. isActive=true when primary mouse button is pressed, isActive=false when released.
isOver	**[isOver: accessType outputOnly, type SFBool (true|false) #FIXED ""]** is pointing device over sensor's geometry?
rotation_changed	**[rotation_changed: accessType outputOnly, type SFRotation CDATA #FIXED ""]** rotation_changed events equal sum of relative bearing changes plus offset value about Y-axis in local coordinate system.
trackPoint_changed	**[trackPoint_changed: accessType outputOnly, type SFVec3f CDATA #FIXED ""]** trackPoint_changed events give intersection point of bearing with sensor's virtual geometry.
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.

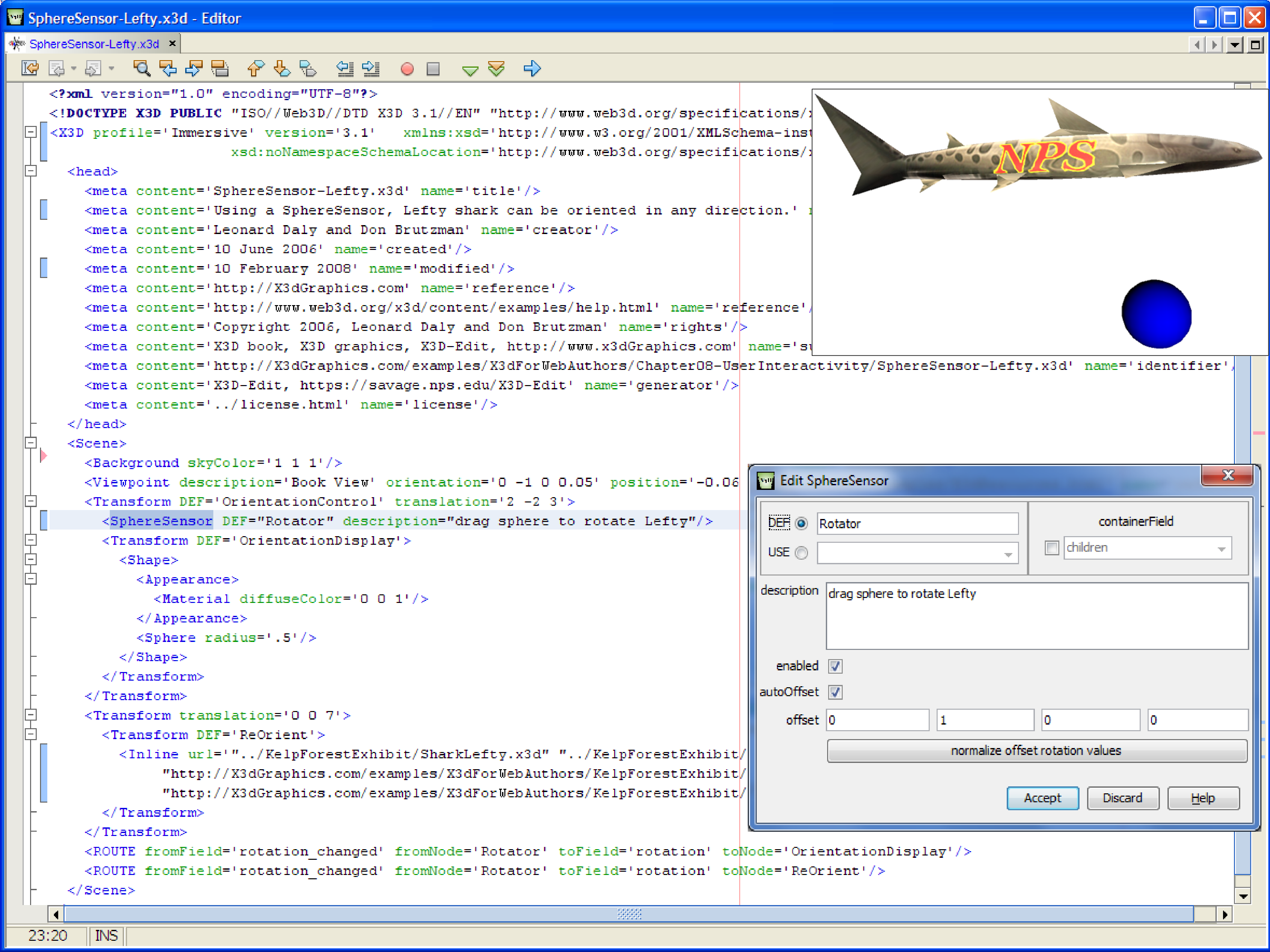
SphereSensor node

- SphereSensor converts x-y dragging motion by the pointing device into an arbitration rotation
  - 2-tuple motion converted to 4-tuple SFRotation
  - Rotation about origin of local coordinate frame
- Activated by peer geometry in scene graph
  - Sensor itself is not rendered, unless corresponding sensed shape itself happens to be spherical
- Rotation output values can have ROUTE connection to parent Transform *rotation* field
  - Or connected to another SFRotation field elsewhere

SphereSensor fields, events

- Sends *isActive* true event when selected
- Sends *isActive* false event when deselected
- *offset* holds latest (or initial) rotation value
- *autoOffset*='true' remembers prior rotation prior to resuming a new drag selection, otherwise *autoOffset*='false' jumps to restart at initial rotation
- *rotation_changed* and *trackPoint_changed* are the basic output events for sensor results

As with all sensors, includes *description*, *enabled*


SphereSensor-Lefty.x3d - Editor
SphereSensor-Lefty.x3d
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-inst
xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/
<head>
<meta content='SphereSensor-Lefty.x3d' name='title'/>
<meta content='Using a SphereSensor, Lefty shark can be oriented in any direction.'
<meta content='Leonard Daly and Don Brutzman' name='creator'/>
<meta content='10 June 2006' name='created'/>
<meta content='10 February 2008' name='modified'/>
<meta content='http://X3dGraphics.com' name='reference'/>
<meta content='http://www.web3d.org/x3d/content/examples/help.html' name='reference'
<meta content='Copyright 2006, Leonard Daly and Don Brutzman' name='rights'/>
<meta content='X3D book, X3D graphics, X3D-Edit, http://www.x3dGraphics.com' name='su
<meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/SphereSensor-Lefty.x3d' name='identifier'
<meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
<meta content='../license.html' name='license'/>
</head>
<Scene>
<Background skyColor='1 1 1'/>
<Viewpoint description='Book View' orientation='0 -1 0 0.05' position='-0.06
<Transform DEF='OrientationControl' translation='2 -2 3'>
<SphereSensor DEF="Rotator" description="drag sphere to rotate Lefty"/>
<Transform DEF='OrientationDisplay'>
<Shape>
<Appearance>
<Material diffuseColor='0 0 1'/>
</Appearance>
<Sphere radius='.5'/>
</Shape>
</Transform>
</Transform>
<Transform translation='0 0 7'>
<Transform DEF='ReOrient'>
<Inline url='"../KelpForestExhibit/SharkLefty.x3d" "../KelpForestExhibit/
"http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/
"http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/
</Transform>
</Transform>
<ROUTE fromField='rotation_changed' fromNode='Rotator' toField='rotation' toNode='OrientationDisplay'/>
<ROUTE fromField='rotation_changed' fromNode='Rotator' toField='rotation' toNode='ReOrient'/>
</Scene>
NPS
Edit SphereSensor
DEF
Rotator
USE
containerField
children
description
drag sphere to rotate Lefty
enabled
autoOffset
offset
0
1
0
0
normalize offset rotation values
Accept
Discard
Help
23:20
INS

SphereSensor	SphereSensor converts pointing device motion into a spherical rotation about the origin of the local coordinate system. Hint: Sensors are affected by peer nodes and children of peers. Hint: add transparent geometry to see the effect of the sensor.
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. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute!
description	**[description: accessType inputOutput, type SFString CDATA #IMPLIED]** Text description to be displayed for action of this node. Hint: use spaces, make descriptions clear and readable. Hint: many XML tools substitute XML character references automatically if needed (like & for & or " for ").
enabled	**[enabled: accessType inputOutput, type SFBool (true|false) "true"]** Enables/disables node operation.
autoOffset	**[autoOffset: accessType inputOutput, type SFBool (true|false) "true"]** Determines whether previous offset values are remembered/accumulated.
offset	**[offset: accessType inputOutput, type SFRotation CDATA "0 1 0 0"]** Sends event and remembers last value sensed.
isActive	**[isActive: accessType outputOnly, type SFBool (true|false) #FIXED ""]** isActive true/false events are sent when triggering the sensor. isActive=true when primary mouse button is pressed, isActive=false when released.
isOver	**[isOver: accessType outputOnly, type SFBool (true|false) #FIXED ""]** is pointing device over sensor's geometry?
rotation_changed	**[rotation_changed: accessType outputOnly, type SFRotation CDATA #FIXED ""]** rotation_changed events equal sum of relative bearing changes plus offset value.
trackPoint_changed	**[trackPoint_changed: accessType outputOnly, type SFVec3f CDATA #FIXED ""]** trackPoint_changed events give intersection point of bearing with sensor's virtual geometry.
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.

KeySensor node

- KeySensor is a one-character-at-a-time interface, capturing key presses from user's keyboard
  - Helpful for selecting from menu choices
  - Helpful for creating a special keyboard-driven navigation interface
  - Only gives key name, not precise shifted character
- Control, alt, shift keys sent as separate events
  - As are certain special "action keys"
- Processing key events requires a Script node
  - Covered in Chapter 9, Event Utilities and Scripting

KeySensor events 1

- Sends *isActive* true event when selected
- Sends *isActive* false event when deselected
- *keyPress*, *keyRelease* provide SFString value for the specific key pressed (or released)
  - Usually upper-case or primary key symbol only
- *shiftKey, altKey, controlKey* are SFBool binary values indicating whether keys were pressed or released

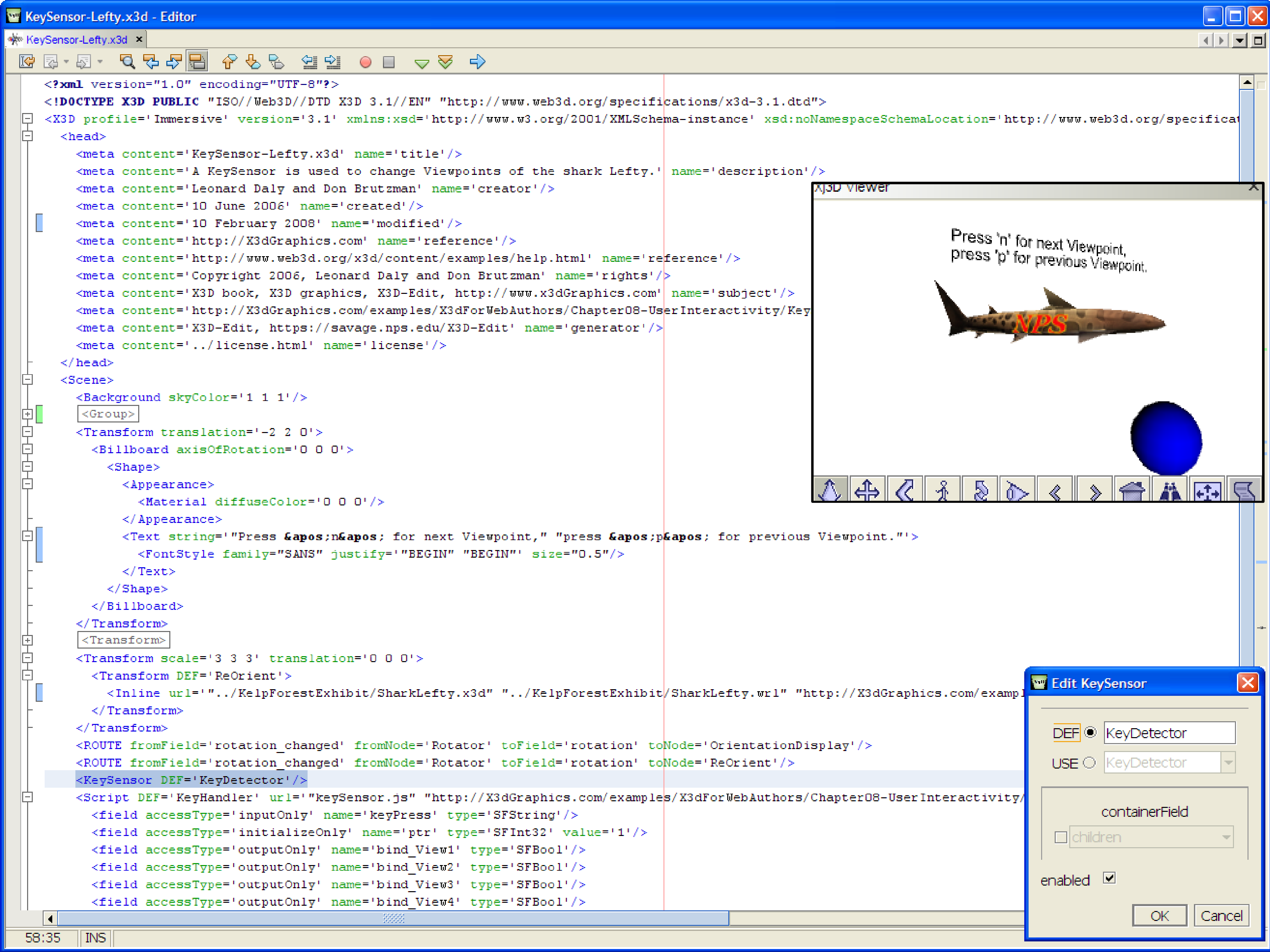
KeySensor also has *enabled* field

- but not *description* since display is challenging

KeySensor events 2

- *actionKeyPress*, *actionKeyRelease* provide SFInt32 values when pressed or released

Key	Value	Interaction Default
F1-F12	1-12	
Home	13	First viewpoint
End	14	Last viewpoint
PageUp	15	Prior viewpoint
PageDown	16	Next viewpoint
Arrow up	17	Cursor up
Arrow down	18	Cursor down
Arrow left	19	Cursor left
Arrow right	20	Cursor right


KeySensor-Lefty.x3d - Editor
KeySensor-Lefty.x3d
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/specificat
<head>
<meta content='KeySensor-Lefty.x3d' name='title'/>
<meta content='A KeySensor is used to change Viewpoints of the shark Lefty.' name='description'/>
<meta content='Leonard Daly and Don Brutzman' name='creator'/>
<meta content='10 June 2006' name='created'/>
<meta content='10 February 2008' name='modified'/>
<meta content='http://X3dGraphics.com' name='reference'/>
<meta content='http://www.web3d.org/x3d/content/examples/help.html' name='reference'/>
<meta content='Copyright 2006, Leonard Daly and Don Brutzman' name='rights'/>
<meta content='X3D book, X3D graphics, X3D-Edit, http://www.x3dGraphics.com' name='subject'/>
<meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/Key
<meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
<meta content='../license.html' name='license'/>
</head>
<Scene>
<Background skyColor='1 1 1'/>
<Group>
<Transform translation='-2 2 0'>
<Billboard axisOfRotation='0 0 0'>
<Shape>
<Appearance>
<Material diffuseColor='0 0 0'/>
</Appearance>
<Text string='"Press 'n' for next Viewpoint," "press 'p' for previous Viewpoint."'>
<FontStyle family="SANS" justify='"BEGIN" "BEGIN"' size="0.5"/>
</Text>
</Shape>
</Billboard>
</Transform>
<Transform>
<Transform scale='3 3 3' translation='0 0 0'>
<Transform DEF='ReOrient'>
<Inline url='"../KelpForestExhibit/SharkLefty.x3d" "../KelpForestExhibit/SharkLefty.wrl" "http://X3dGraphics.com/examp
</Transform>
</Transform>
<ROUTE fromField='rotation_changed' fromNode='Rotator' toField='rotation' toNode='OrientationDisplay'/>
<ROUTE fromField='rotation_changed' fromNode='Rotator' toField='rotation' toNode='ReOrient'/>
<KeySensor DEF='KeyDetector'/>
<Script DEF='KeyHandler' url='"keySensor.js" "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/
<field accessType='inputOnly' name='keyPress' type='SFString'/>
<field accessType='outputOnly' name='bind_View1' type='SFBool'/>
<field accessType='outputOnly' name='bind_View2' type='SFBool'/>
<field accessType='outputOnly' name='bind_View3' type='SFBool'/>
<field accessType='outputOnly' name='bind_View4' type='SFBool'/>
58:35
INS
Xj3D Viewer
Press 'n' for next Viewpoint,
press 'p' for previous Viewpoint.
NPS
Edit KeySensor
DEF
KeyDetector
USE
KeyDetector
containerField
children
enabled
OK
Cancel

keySensor.js - Editor

keySensor.js

```
// Description: Process key presses to bind next, previous viewpoints
// Filename:    keySensor.js
// Author:      Len Daly and Don Brutzman
// Identifier:  http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/keySensor.js
// Created:     10 June 2006
// Revised:     18 February 2008
// Reference:   http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/KeySensor-Lefty.x3d
// License:     http://X3dGraphics.com/examples/X3dForWebAuthors/license.html

function keyPress (value) {
  Browser.print ('Key press = >' + value + '< Initial pointer: ' + ptr + '\n');
  // bind next viewpoint
  if (value == 'N' || value == 'n') {
    ptr ++;
    if (ptr > 6) {
      ptr = 1;
    }
    if (ptr == 1) {
      bind_View1 = true;
    }
    if (ptr == 2) {
      bind_View2 = true;
    }
    if (ptr == 3) {
      bind_View3 = true;
    }
    if (ptr == 4) {
      bind_View4 = true;
    }
    if (ptr == 5) {
      bind_View5 = true;
    }
    if (ptr == 6) {
      bind_View6 = true;
    }
  }
  // similarly, bind previous viewpoint
  if (value == 'P' || value == 'p') {
    if (ptr == 1) {
      bind_View1 = false;
    }
    if (ptr == 2) {
```

2:29 INS

KeySensor	KeySensor generates events as the user presses keys on the keyboard. Supports notion of "keyboard focus".
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. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute!
enabled	**[enabled: accessType inputOutput, type SFBool (true|false) "true"]** Enables/disables node operation.
keyPress	**[keyPress: accessType outputOnly, type SFString CDATA #IMPLIED]** Events generated when user presses character-producing keys on keyboard produces integer UTF-8 character values.
keyRelease	**[keyRelease: accessType outputOnly, type SFString CDATA #IMPLIED]** Events generated when user releases character-producing keys on keyboard produces integer UTF-8 character values.
actionKeyPress	**[actionKeyPress: accessType outputOnly, type SFInt32 CDATA #IMPLIED]** action key press gives following values: HOME=000 END=1001 PGUP=1002 PGDN=1003 UP=1004 DOWN=1005 LEFT=1006 RIGHT=1007 F1..F12 = 1008..1019.
actionKeyRelease	**[actionKeyRelease: accessType outputOnly, type SFInt32 CDATA #IMPLIED]** action key release gives following values: HOME=000 END=1001 PGUP=1002 PGDN=1003 UP=1004 DOWN=1005 LEFT=1006 RIGHT=1007 F1..F12 = 1008..1019.
shiftKey	**[shiftKey: accessType outputOnly, type SFBool (true|false) #IMPLIED]** shiftKey generates true event when pressed, false event when released.
controlKey	**[controlKey: accessType outputOnly, type SFBool (true|false) #IMPLIED]** controlKey generates true event when pressed, false event when released.
altKey	**[altKey: accessType outputOnly, type SFBool (true|false) #IMPLIED]** altKey generates true event when pressed, false event when released.
isActive	**[isActive: accessType outputOnly, type SFBool (true|false) #FIXED ""]** isActive true/false events are sent when triggering the sensor. isActive=true when primary mouse button is pressed, isActive=false when released.
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.

StringSensor node, events

StringSensor provides a string-based interface to the user's keyboard

- Each character key press is collected until <Enter> key is returned, completing *finalText* string
- Intermediate string results (including deletions) also available as user proceeds in *enteredText* string
- *deletionAllowed* is boolean field that enables <Backspace>, <Delete> keys

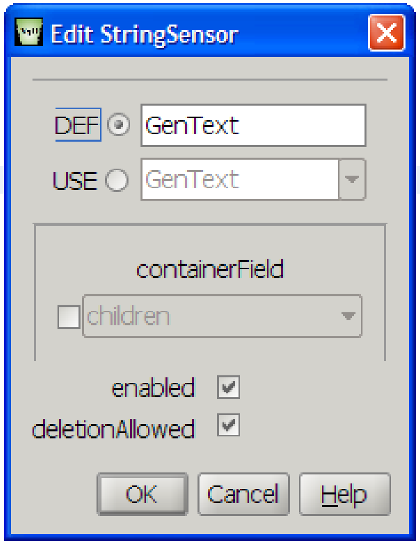
StringSensor has *isActive* events, *enabled* field

- but not *description* since display is challenging

StringSensor.x3d - Editor

StringSensor.x3d

```
  <head>
    <meta content='StringSensor.x3d' name='title'/>
    <meta content='A StringSensor example that displays typed text in the world.' name='description'/>
    <meta content='Leonard Daly and Don Brutzman' name='creator'/>
    <meta content='7 June 2006' name='created'/>
    <meta content='18 February 2008' name='modified'/>
    <meta content='http://X3dGraphics.com' name='reference'/>
    <meta content='http://www.web3d.org/x3d/content/examples/help.html' name='reference'/>
    <meta content='Copyright (c) 2006, Daly Realism and Don Brutzman' name='rights'/>
    <meta content='X3D book, X3D graphics, X3D-Edit, http://www.x3dGraphics.com' name='subject'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/StringSensor.x3d' name='identifier'/>
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <Background skyColor='1 1 1'/>
    <Viewpoint description='Book View' position='-0.02 0.01 6.85'/>
    <StringSensor DEF='GenText' deletionAllowed='true' enabled='true'/>
    <Transform>
      <Transform translation='0 0 -.1'>
        <Shape>
          <Appearance>
            <Material diffuseColor='1 1 .6'/>
          </Appearance>
          <Box size='8 1.5 .01'/>
        </Shape>
      </Transform>
      <Transform translation='-3.8 0.2 0'>
        <Shape>
          <Appearance>
            <Material diffuseColor='0 0 1'/>
          </Appearance>
          <Text DEF='DisplayText'>
            <FontStyle justify='"BEGIN" "MIDDLE"' size="0.75"/>
          </Text>
        </Shape>
      </Transform>
      <Script DEF='Converter' url='"converter.js" "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/converter.js"'>
        <field accessType='inputOnly' name='SFString_MFString' type='SFString'/>
        <field accessType='outputOnly' name='MFString_out' type='MFString'/>
      </Script>
      <ROUTE fromField='enteredText' fromNode='GenText' toField='SFString_MFString' toNode='Converter'/>
      <ROUTE fromField='MFString_out' fromNode='Converter' toField='string' toNode='DisplayText'/>
    </Transform>
  </Scene>
</X3D>
```



Type text using StringSensor

21:18 INS

converter.js - Editor

converter.js

```
// Description: Collection of various data-type conversion utility methods.
// Filename:    converter.js
// Author:      Len Daly and Don Brutzman
// Identifier:  http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/converter.js
// Created:     17 June 2006
// Revised:     18 February 2008
// Reference:   http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/StringSensor.x3d
// License:     http://X3dGraphics.com/examples/X3dForWebAuthors/license.html
//
//      The name of a method indicates the incoming and outgoing datatypes.
//
//      If a particular element needs to be selected (e.g., SFVec3F to SFFloat), then
//      the element name is indicated after the incoming datatype (e.g., SFVec3fX means
//      the X ([0]) element of the datum.
//
//      Outgoing values (events) are always named after the datatype with '_out' appended.
//      If a particular element was selected, then the element name appears after the
//      underscore and before 'out'.
//
//      The exception to this naming convention is a conversion from MF* to SF*. In that
//      case, the first element ([0]) is always taken and no special notation is used.

function MFString_SFString (value) {
  SFString_out = value[0];
}

function SFString_MFString (value) {
  MFString_out = new MFString (value);
}

function SFVec3fX_SFFloat (value) {
  SFFloat_Xout = value[0];
}

function SFVec3fY_SFFloat (value) {
  SFFloat_Yout = value[1];
}

function SFVec3fZ_SFFloat (value) {
  SFFloat_Zout = value[2];
}
```

27:27 INS

StringSensor	StringSensor generates events as the user presses keys on the keyboard.
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. **Warning:** do NOT include DEF (or any other attribute values) when using a USE attribute!
enabled	**[enabled: accessType inputOutput, type SFBool (true|false) "true"]** Enables/disables node operation.
deletionAllowed	**[deletionAllowed: accessType inputOutput, type SFBool (true|false) "true"]** If deletionAllowed is true, then previously entered character in enteredText can be removed. If deletionAllowed is false, then characters may only be added to the string. **Hint:** deletion key is typically defined by local system.
isActive	**[isActive: accessType outputOnly, type SFBool (true|false) #FIXED ""]** isActive true/false events are sent when triggering the sensor. isActive=true when primary mouse button is pressed, isActive=false when released.
enteredText	**[enteredText: accessType outputOnly, type SFString CDATA #FIXED ""]** Events generated as character-producing keys are pressed on keyboard.
finalText	**[finalText: accessType outputOnly, type SFString CDATA #FIXED ""]** Events generated when sequence of keystrokes matches keys in terminationText string when this condition occurs, enteredText is moved to finalText and enteredText is set to empty string. **Hint:** termination key is typically defined by local system.
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.

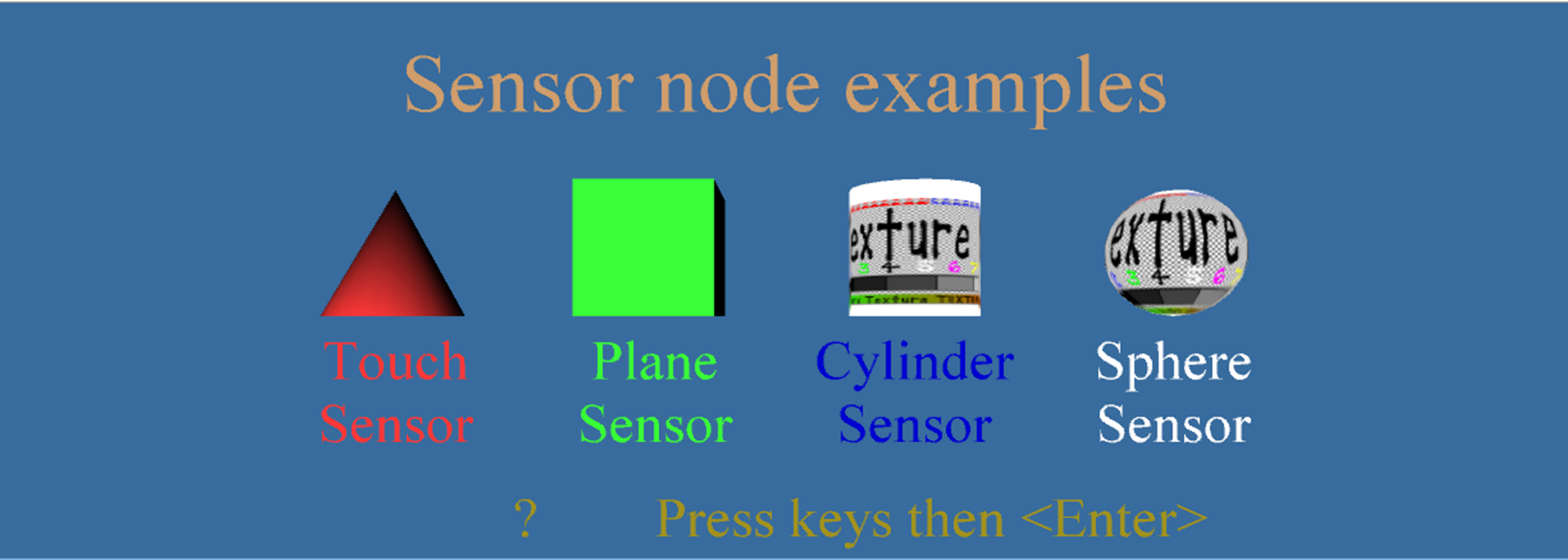
Example: user-interactivity sensor nodes

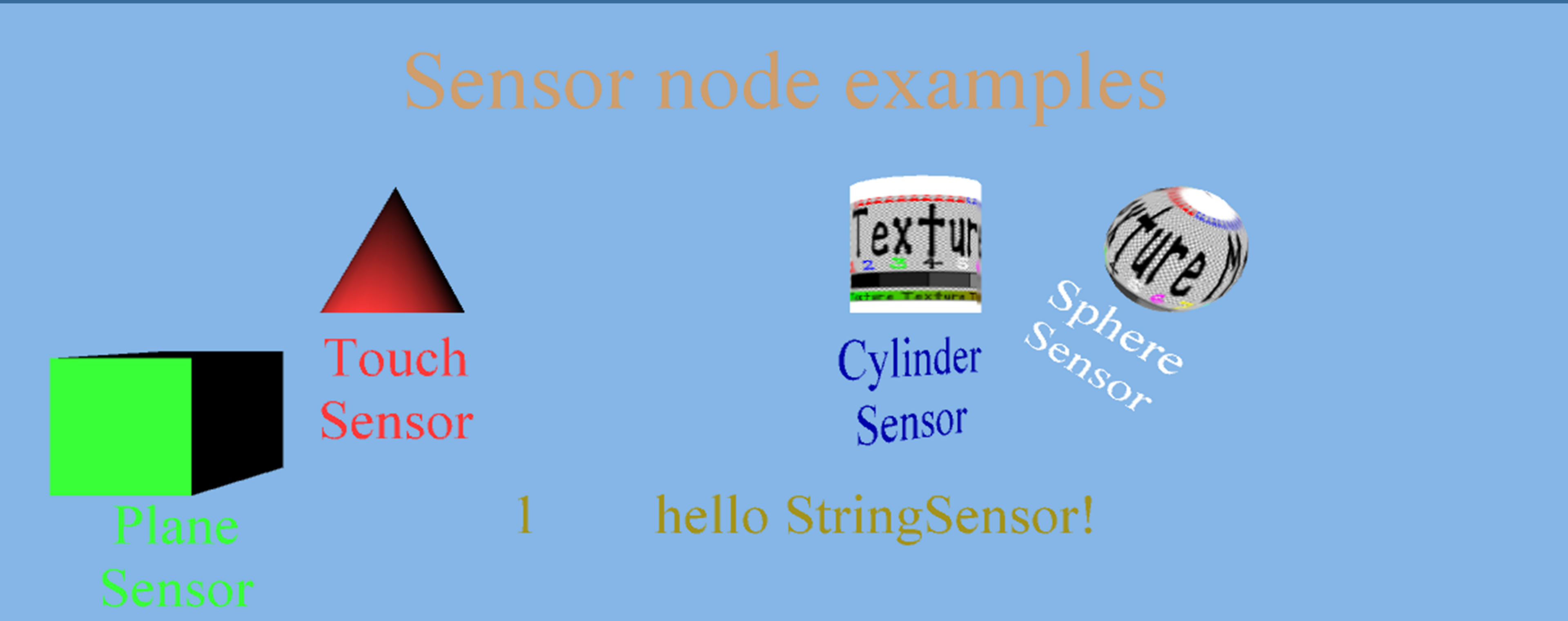
UserInteractivitySensorNodes.x3d

- Select (click and hold) TouchSensor Cone to alternate Background nodes
- Select and drag PlaneSensor -- Box on the screen
- Select and drag to rotate CylinderSensor -- Cylinder
- Select and drag to spin SphereSensor -- Sphere

Keyboard inputs are also activated

- KeySensor indicates keyPress
- StringSensor shows *finalText* once <Enter> pressed
- Console shows *enteredText* (includes deletes if any)


Sensor node examples
Touch Sensor
Plane Sensor
Cylinder Sensor
Sphere Sensor
? Press keys then <Enter>


Sensor node examples
Touch Sensor
Cylinder Sensor
Sphere Sensor
Plane Sensor
1 hello StringSensor!

UserInteractivitySensorNodes.x3d - Editor

UserInteractivitySensorNodes.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.
  <head>
    <meta content='UserInteractivitySensorNodes.x3d' name='title'/>
    <meta content='A collection of all of the user interactivity sensor nodes: TouchSensor, PlaceSensor, CylinderSensor, SphereSensor, KeySen
    <meta content='Don Brutzman' name='creator'/>
    <meta content='30 April 2005' name='created'/>
    <meta content='16 February 2008' name='modified'/>
    <meta content='Copyright 2006, Daly Realism and Don Brutzman' name='rights'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/UserInteractivitySensorNodes.x3d' name='ident
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <Viewpoint description='User interactivity sensor nodes' position='0 0 12'/>
    <Background DEF='BackgroundDefault' groundColor='0.2 0.4 0.6' skyColor='0.2 0.4 0.6'/>
    <Background DEF='BackgroundTouchCone' skyColor='0.5 0.7 0.9'/>
    <Transform translation='0 4 0'>
      <Shape>
        <Text string='Sensor node examples'>
          <FontStyle justify='"MIDDLE" "MIDDLE"' size='1.5'/>
        </Text>
        <Appearance>
          <Material DEF='DefaultMaterial' diffuseColor='0.8 0.6 0.4'/>
        </Appearance>
      </Shape>
    </Transform>
    <Transform translation='0 1 0'>
      <Transform translation='-6 0 0'>
        <TouchSensor DEF='DefaultTouchSensor'  description='click to activate TouchSensor bind alternate Background' enabled='true'/>
        <Shape>
          <Cone/>
          <Appearance DEF="RedAppearance">
            <Material diffuseColor='1 0.2 0.2'/>
          </Appearance>
        </Shape>
        <Transform translation="0 -2 0">
            <Shape>
                    <Text string='"Touch" "Sensor"' solid="false">
                            <FontStyle DEF="JustifyMiddle" justify='"MIDDLE" "MIDDLE"'/>
                    </Text>
                    <Appearance USE="RedAppearance"/>
            </Shape>
        </Transform>
```

31:21 INS

UserInteractivitySensorNodes.x3d - Editor

UserInteractivitySensorNodes.x3d

```
      <Transform DEF='TransformBox' translation='-2 0 0'>
        <PlaneSensor DEF="DefaultPlaneSensor" description="drag Box to activate PlaneSensor"/>
        <Shape>
          <Box/>
          <Appearance>
        </Shape>
        <Transform translation="0 -2 0">
            <Shape>
                    <Text string='"Plane" "Sensor"' solid="false">
                            <FontStyle USE="JustifyMiddle"/>
                    </Text>
                    <Appearance USE="GreenAppearance"/>
            </Shape>
        </Transform>
        <ROUTE fromField='offset' fromNode='DefaultPlaneSensor' toField='set_translation' toNode='TransformBox'/>
      </Transform>
      <Transform DEF='TransformCylinder' translation='2 0 0'>
        <CylinderSensor DEF="DefaultCylinderSensor" description="drag to activate CylinderSensor"/>
        <Shape>
          <Cylinder/>
          <Appearance>
            <ImageTexture DEF='ReferenceTexture' url='"../Chapter05-AppearanceMaterialTextures/ImageTextureFigure18.1X3dSpecification.gif"
            "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter05-AppearanceMaterialTextures/ImageTextureFigure18.1X3dSpecification.gif"'/>
          </Appearance>
        </Shape>
        <Transform translation="0 -2 0">
            <Shape>
                    <Text string='"Cylinder" "Sensor"' solid="false">
                            <FontStyle USE="JustifyMiddle"/>
                    </Text>
                    <Appearance>
            </Shape>
        </Transform>
        <ROUTE fromField='rotation_changed' fromNode='DefaultCylinderSensor' toField='set_rotation' toNode='TransformCylinder'/>
      </Transform>
      <Transform DEF='TransformSphere' translation='6 0 0'>
        <SphereSensor DEF="DefaultSphereSensor" description="click to activate SphereSensor"/>
        <Shape>
          <Sphere/>
          <Appearance>
            <ImageTexture USE='ReferenceTexture'/>
          </Appearance>
        </Shape>
        <Transform DEF='SphereSensorText' translation="0 -2 0">
            <Shape>
        </Transform>
```

67:24 INS

UserInteractivitySensorNodes.x3d - Editor

UserInteractivitySensorNodes.x3d

```
        <Shape>
                <Text DEF='KeyText' solid="false" string='?'>
                        <FontStyle USE="JustifyMiddle"/>
                </Text> <Appearance>
        </Shape>
      </Transform>
      <Transform translation="-2 -3 0">
        <Shape>
                <Text DEF='StringText' solid="false" string='Press keys then &lt;Enter&gt;'>
                        <FontStyle justify='"BEGIN" "MIDDLE"'/>
                </Text> <Appearance USE="BrownAppearance"/>
        </Shape>
      </Transform>
    <KeySensor DEF='DefaultKeySensor'  enabled='true'/>
    <StringSensor DEF='DefaultStringSensor'  deletionAllowed='true' enabled='true'/>
    <Script DEF="KeyboardProcessor">
      <field name='keyInput' type='SFString' accessType='inputOnly'/>
      <field name='finalTextInput' type='SFString' accessType='inputOnly'/>
      <field name='enteredTextInput' type='SFString' accessType='inputOnly'/>
      <field name='keyOutput' type='MFString' accessType='outputOnly'/>
      <field name='stringOutput' type='MFString' accessType='outputOnly'/>
<![CDATA[
ecmascript:

function keyInput (inputValue)
{
    Browser.print ('keyInput=' + inputValue + '\n'); // console output
    keyOutput = new MFString (inputValue); // type conversion
}
function finalTextInput (inputValue)
{
    Browser.print ('finalText=' + inputValue + '\n'); // console output
    stringOutput = new MFString (inputValue); // type conversion
}
function enteredTextInput (inputValue)
{
    Browser.print ('enteredText=' + inputValue + '\n'); // console output
}
]]>
      </Script>
      <ROUTE fromNode='DefaultKeySensor' fromField='keyPress' toNode='KeyboardProcessor' toField='keyInput'/>
      <ROUTE fromNode='DefaultStringSensor' fromField='finalText' toNode='KeyboardProcessor' toField='finalTextInput'/>
      <ROUTE fromNode='DefaultStringSensor' fromField='enteredText' toNode='KeyboardProcessor' toField='enteredTextInput'/>
      <ROUTE fromNode='KeyboardProcessor' fromField='keyOutput' toNode='KeyText' toField='string'/>
      <ROUTE fromNode='KeyboardProcessor' fromField='stringOutput' toNode='StringText' toField='string'/>
```

123:1 INS

back to Table of Contents

Chapter Summary

Summary: User Interactivity

User interactivity is initiated via sensor nodes, which capture user inputs and are hooked up to provide appropriate responses

- TouchSensor senses pointing device (mouse, etc.)
- PlaneSensor is a drag sensor that converts x-y pointer motion to move objects in a plane
- CylinderSensor and SphereSensor are drag sensors that convert x-y pointer motion to rotate objects
- KeySensor and StringSensor capture keyboard input

Interactivity sensors initiate animation chains

Suggested exercises

Illustrate and annotate ROUTE connections in an animation scene graph (documenting 10 steps)

- Print out one of these scenes in landscape mode, either using the X3dToXhtml.xslt stylesheet version or Netbeans-provided 'Save as HTML' option.
- Then draw all ROUTE connections, label beginning and end of each by name, type and accessType
- Best candidate: UserInteractivitySensorNodes.x3d

Draw animation chain diagrams to document behaviors in your own example scenes

- Add use-case summaries about user intent

Additional Resources

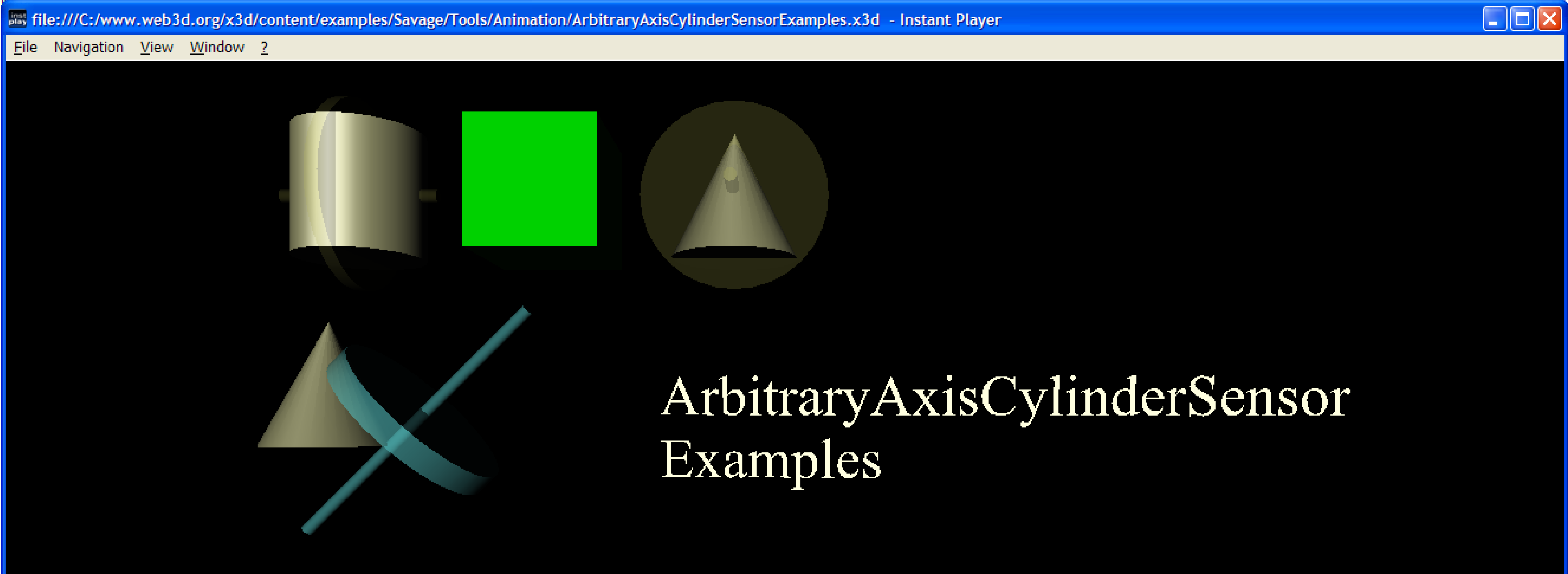
ArbitraryAxisCylinderSensor Prototype

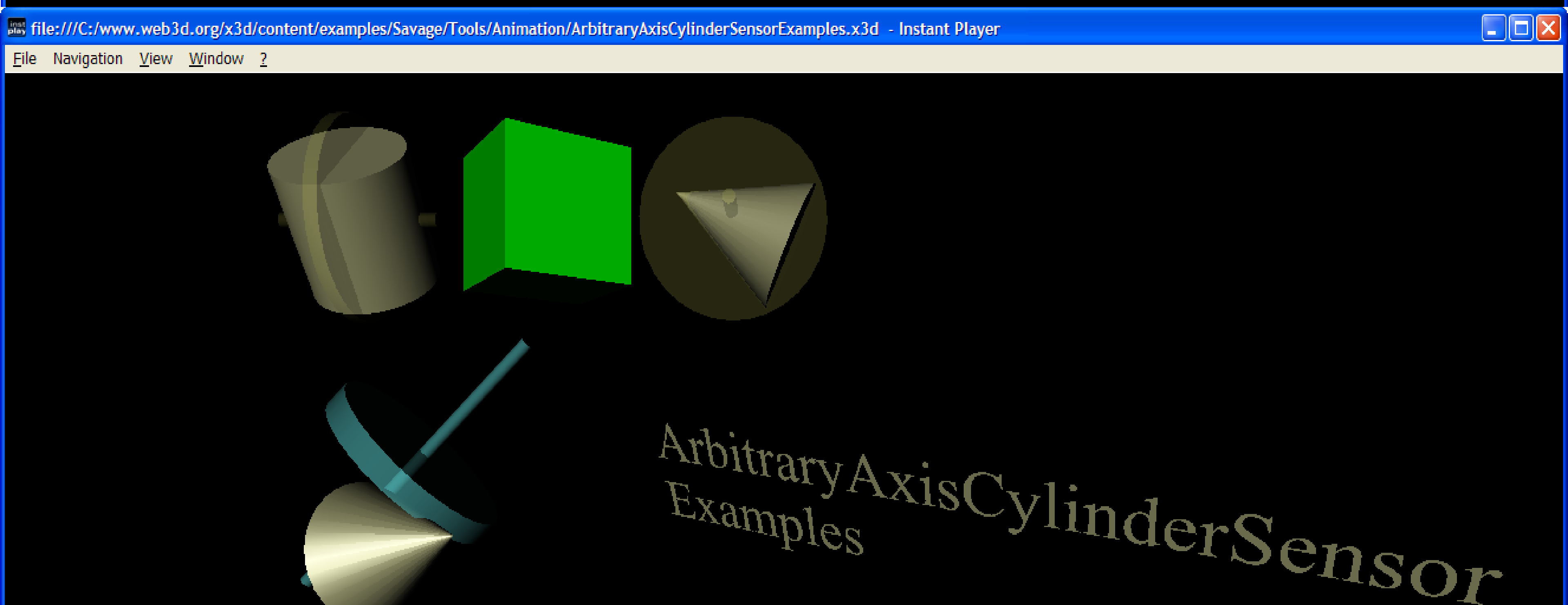
ArbitraryAxisCylinderSensor is a prototype that simplifies the design pattern of aligning a CylinderSensor about an arbitrary axis

- https://savage.nps.edu/Savage/Tools/Animation
- Prototype definition: ArbitraryAxisCylinderSensorPrototype.x3d
- ProtoInstance examples: ArbitraryAxisCylinderSensorExamples.x3d

Fields match those of CylinderSensor, plus:

- *shiftRotationAxis, center, children*, plus show/scale/color/transparency of CylinderSensorShape


file:///C:/www.web3d.org/x3d/content/examples/Savage/Tools/Animation/ArbitraryAxisCylinderSensorExamples.x3d - Instant Player
File Navigation View Window ?
ArbitraryAxisCylinderSensor
Examples


file:///C:/www.web3d.org/x3d/content/examples/Savage/Tools/Animation/ArbitraryAxisCylinderSensorExamples.x3d - Instant Player
File Navigation View Window ?
ArbitraryAxisCylinderSensor
Examples

DoubleClickTouchSensor

DoubleClickTouchSensor is a prototype alternative to TouchSensor that detects when a user has rapidly selected an object twice

- https://savage.nps.edu/Savage/Tools/Animation
- Prototype definition: DoubleClickTouchSensorPrototype.x3d
- ProtoInstance examples: DoubleClickTouchSensorExample.x3d

Fields match those of TouchSensor, plus:

- *maxDelayInterval* allowed for distinguishing between single and double click, in seconds

TimeDelaySensor Prototype

TimeDelaySensor is an alternative to TimeSensor that includes a time delay before firing

- https://savage.nps.edu/Savage/Tools/Animation
- Prototype definition: TimeDelaySensorPrototype.x3d
- ProtoInstance examples: TimeDelaySensorExample.x3d

Fields match those of TimeSensor, plus:

- *delayInterval, delayCompleteTime*

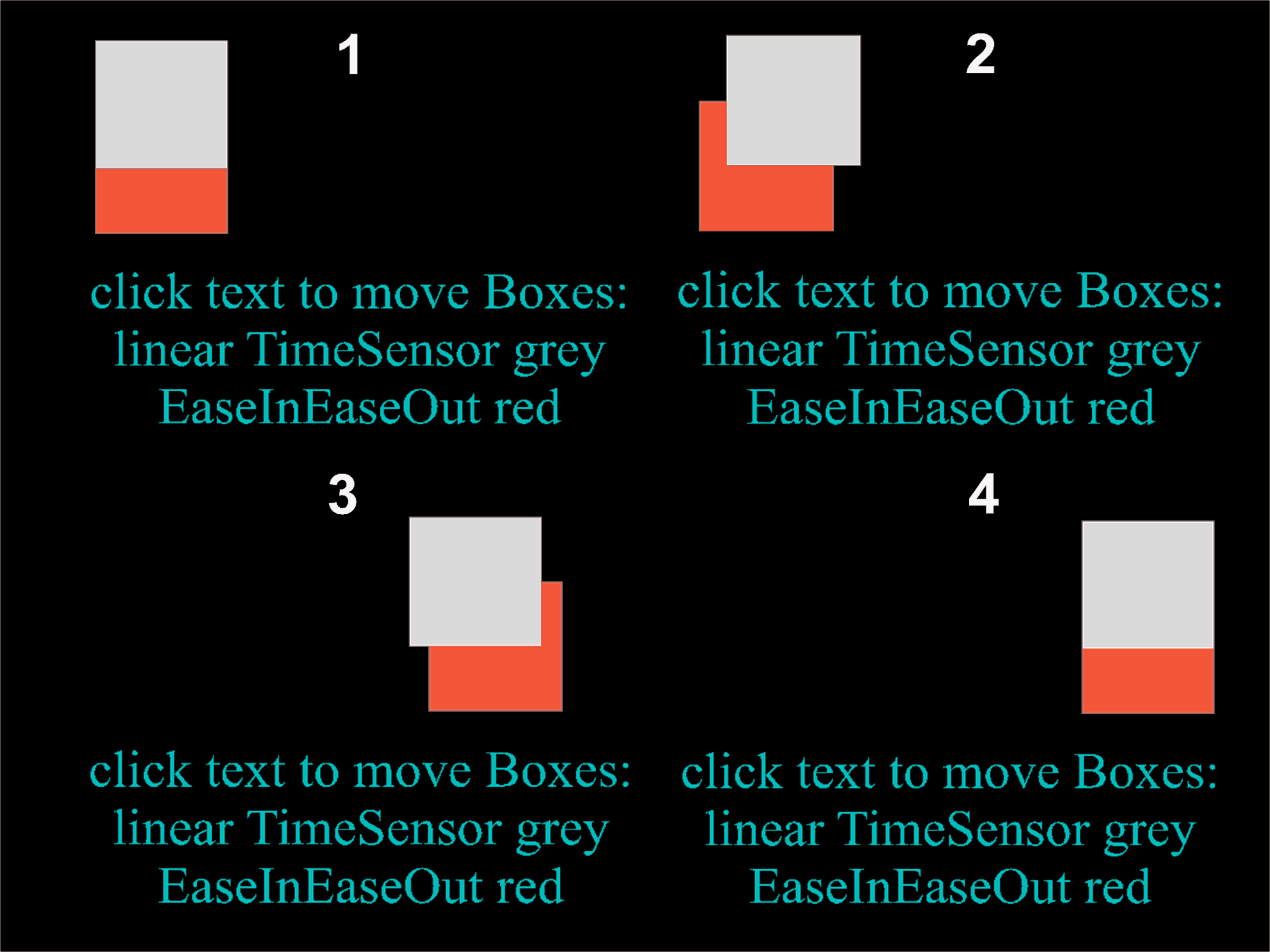
TimeSensorEaseInEaseOut Prototype

TimeSensorEaseInEaseOut is an alternative to TimeSensor with a slower ramp at beginning and end of a cycle, thus smoothing transitions

- https://savage.nps.edu/Savage/Tools/Animation
- Prototype definition: TimeSensorEaseInEaseOutPrototype.x3d
- ProtoInstance examples: TimeSensorEaseInEaseOutExample.x3d

Fields match those of TimeSensor

- Slight linear slowdown for first and last 10%
- Slight linear speedup in between

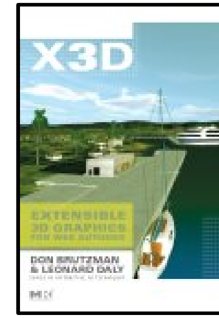

1
click text to move Boxes:
linear TimeSensor grey
EaseInEaseOut red
2
click text to move Boxes:
linear TimeSensor grey
EaseInEaseOut red
3
click text to move Boxes:
linear TimeSensor grey
EaseInEaseOut red
4
click text to move Boxes:
linear TimeSensor grey
EaseInEaseOut red

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 8, User Interactivity
- http://x3dGraphics.com
- http://x3dgraphics.com/examples/X3dForWebAuthors

X3D Resources

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

web3D CONSORTIUM

<X3D>

References 2

X3D-Edit Authoring Tool

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

X3D Scene Authoring Hints

- http://x3dgraphics.com/examples/X3dSceneAuthoringHints.html

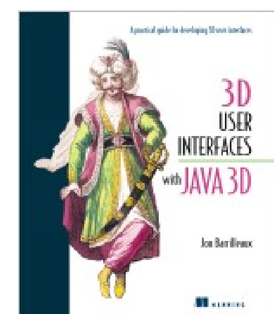
X3D Graphics Specification

- 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 9 - Sensing Viewer

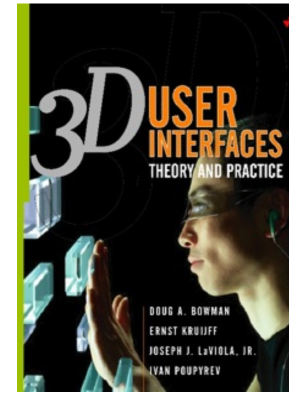


3D User Interfaces with Java3D by Jon Barilleaux, Manning Publications, 2001.

- http://www.manning.com/barrilleaux
- http://java.sun.com/developer/Books/Java3D

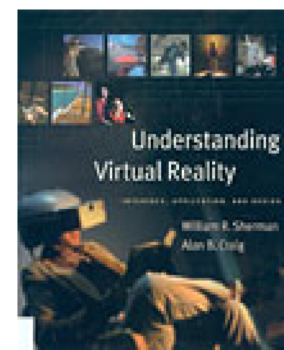
References 4

3D User Interfaces: Theory and Practice by Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola Jr. and Ivan Poupyrev, Addison Wesley, 2005.



- http://www.3dui.org
- http://people.cs.vt.edu/~bowman/3dui.org/3D UI Book.html

Understanding Virtual Reality: Interface, Application and Design by Bill Sherman and Alan Craig, Morgan Kaufmann, 2003.



- http://www.immersence.com/publications/2003/2003-WSherman.html

web3D CONSORTIUM

Conferences 1

ACM SIGGRAPH

- Special Interest Group on Graphics is the leading professional society for computer graphics and interactive techniques
- http://www.siggraph.org

ACM SIGCHI

- Special Interest Group on Computer-Human Interaction, brings together people working on the design, evaluation, implementation, and study of interactive computing systems for human use
- http://www.sigchi.org

web3D CONSORTIUM

<X3D>

Conferences 2

IEEE Symposium on 3D User Interfaces (3DUI)

- http://conferences.computer.org/3dui

IEEE Symposium on Virtual Reality (VR)

- http://conferences.computer.org/vr

Web3D Symposium

- In cooperation with Web3D Consortium, ACM SIGGRAPH and Eurographics
- http://www.web3d2009.org

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

web3D 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

Creative Commons open-source license

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

creative commons

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

X3D Graphics for Web Authors

Chapter 8

User Interactivity

Nobody knows the kind of trouble we're in.
Nobody seems to think it all might happen again.
Gram Parsons, "One Hundred Years from Now"

web3D CONSORTIUM

1

Listen to one of the first (and perhaps still greatest) country rock albums of all time by the Byrds: Sweetheart of the Rodeo, 1968. The cover is by Monterey California artist Jo Mora and was used for the Salinas Rodeo. "One Hundred Years from Now" is among Gram Parson's best songs.

One Hundred Years from Now by Gram Parsons

One hundred years from this day
Will the people still feel this way
Still say the things that they're saying right now?
Everyone said I'd hurt you
They said I'd desert you
If I go away, you know I'm gonna get back somehow

Nobody knows what kind of trouble we're in
Nobody seems to think it all might happen again

One hundred years from this time
Would anybody change their mind
And find out one thing or two about life?
But people are always talking
You know they're always talking
Everybody's so wrong that I know it's gonna work out right

It's fun to take the long view when thinking about X3D.

Contents

web3D CONSORTIUM

2

Chapter Overview

3

Overview: User Interactivity

User interactivity is initiated via sensor nodes, which capture user inputs and are hooked up to provide appropriate responses

- TouchSensor senses pointing device (mouse, etc.)
- PlaneSensor is a drag sensor that converts x-y pointer motion to move objects in a plane
- CylinderSensor and SphereSensor are drag sensors that convert x-y pointer motion to rotate objects
- KeySensor and StringSensor capture keyboard input

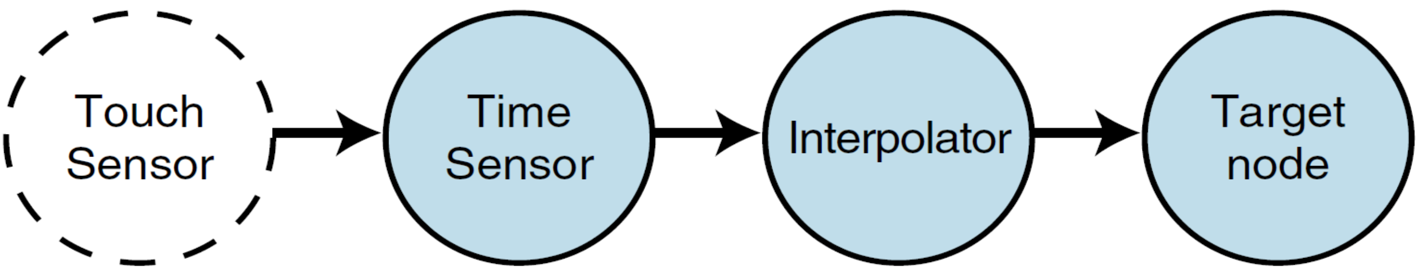
Interactivity sensors initiate animation chains

web 3D CONSORTIUM

4

Dragging is the movement of a selected object using the pointing device, a capability provided by the drag sensors.

Animation chains are covered in Chapter 7, Event Animation and Interpolation.



Related nodes

Chapter 4, Viewing and Navigation nodes

- Anchor: pointing device
  - Selects another Viewpoint or loads another scene
  - Show description when pointing device is over geometry
- Billboard rotates child geometry to face user
- Collision reports if viewer collides with geometry

Chapter 12, Environment Sensor and Sound

- LoadSensor reports when media asset is loaded
- ProximitySensor reports when user is in vicinity
- VisibilitySensor indicates when user's current camera view can see sensed geometry

web|3D CONSORTIUM

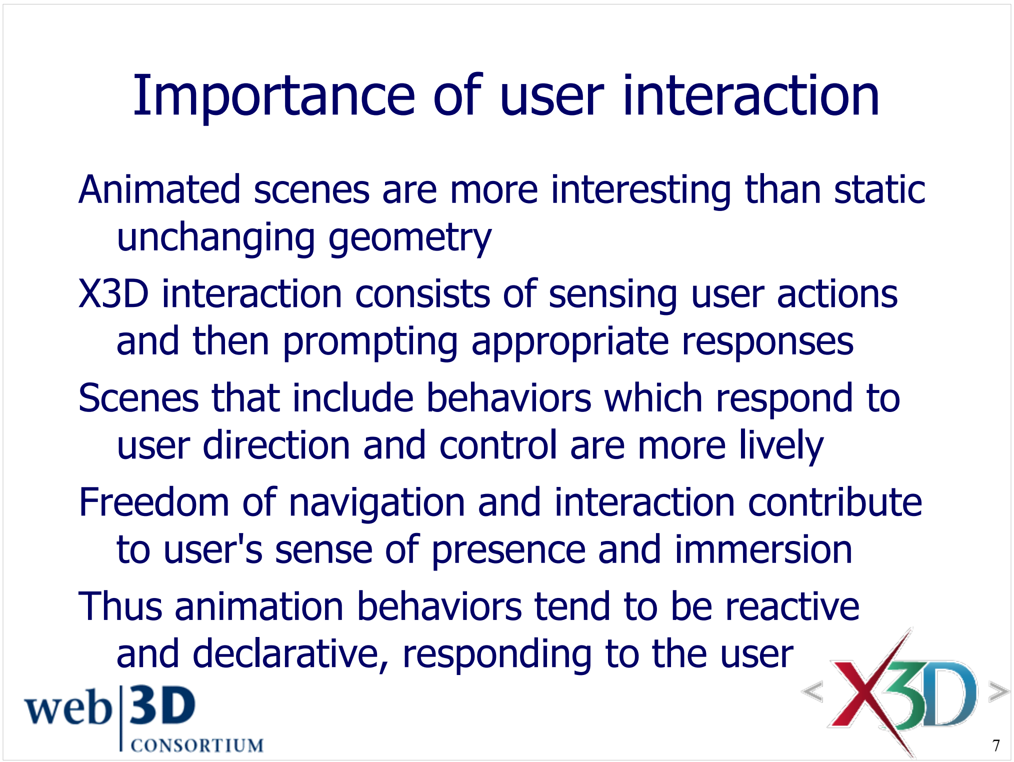
5

User Interactivity nodes directly follow the event model presented in Chapter 7.

back to Table of Contents

Concepts

6



There is a large body of work in 3D user interaction. See the Additional Resources section.

Sensors produce events

Sensors detect various kinds of user interaction and produce events to ROUTE within a scene

- Each sensor detects a certain kind of interaction, then produces one or more events

Authors decide how the events describing user interaction are interpreted and handled

- This approach allows great flexibility for authors

web|3D CONSORTIUM

<X3D>

8

TODO Add route diagram here...

Using sensors in a scene

Three common design patterns (→ = ROUTE)

- Trigger (sensor) → Clock → Interpolator → Target node
- Sensor → Target node
- Sensor → Script (adaptor) → Target node

9

TODO add figure

Pointing devices

Pointing device is primary tool for user interaction with geometry in a scene

- Mouse, Touchpad, touchscreen, or tracking stylus
- Arrow, Enter, other keys are allowed substitutes
- Trackball, data glove, game controller
- Tracking wand or other device in immersive 3D environments (such as a cave)
- Eye trackers and other advanced devices possible

X3D sensors designed for use with any generic pointing device, thus making scenes portable

web 3D CONSORTIUM

<X3D>

10

Very different from most programming approaches...

Sensed geometry intersection, selection

Pointing devices communicate user's intended direction, movement, and selection (if any)

- Browsers and viewers usually superimpose 2D icon to indicate user's intended pointing direction
- 2D overlay icon may change to indicate selection

Sensors react to corresponding sibling and child geometry in the scene graph

- Pointing at other geometry means sensor activation no longer possible
- Usually one sensor must be deactivated before another can become active

web3D CONSORTIUM

Common field: *enabled*

enabled is an inputOutput boolean field that turns a sensor node on or off

- Thus allowing author to permit or disable flow of user-driven events which drive other responses
- Set *enabled*='false' to disrupt an event chain

Regardless of whether *enabled*='true' a sensor still needs a ROUTE connection from its output, or else no interaction response occurs

web3D CONSORTIUM

X3D

12

For author: Get ready...

Common field: *isOver*

isOver is an outputOnly boolean field that reports when pointing at sensed geometry

- *isOver* true value sent when pointer is over shapes
- isOver false value sent when pointer icon is no longer over shapes
- If selection occurred, isOver false doesn't occur until after selection is released

Routing *isOver* values can enable animation

- Rapid sequencing on/off might remain a difficulty
- Take care that animation doesn't move viewpoint or geometry out from under the pointing device

web3D CONSORTIUM

X3D

13

For user: Get set...

Common field: *isActive*

isActive is an outputOnly boolean field that reports when sensor has received user input

- *isActive* true value sent when selection begins
- *isActive* false value sent when selection released
- Note that *isActive* true already occurs as a prerequisite when a sensor is initially enabled

Routing *isActive* values can enable, disable TimeSensor and other animation nodes

- Rapid sequencing on/off can be a difficulty, however
- BooleanFilter, BooleanToggle, BooleanTrigger also useful: Chapter 9 Event Utilities and Scripting

web3D CONSORTIUM

For user: Go!!

Common field: *description*

Each sensor's *description* field alerts users to the presence and intended purpose of each sensor

- Thus including a *description* is quite important, otherwise user is left to guess about responses
- Nevertheless many authors forget to include *description,* which inhibits interactivity

X3D Specification gives browsers flexibility about how *description* strings are displayed

- Overlay text, window-border text, perhaps audio

web 3D CONSORTIUM

15

Dragging

Dragging means to select (activate) a sensed object, then to move the pointing device while the sensor is still activated

This user action causes continuous generation of output events while dragging motion occurs

- Click + drag + release = Select + hold + release

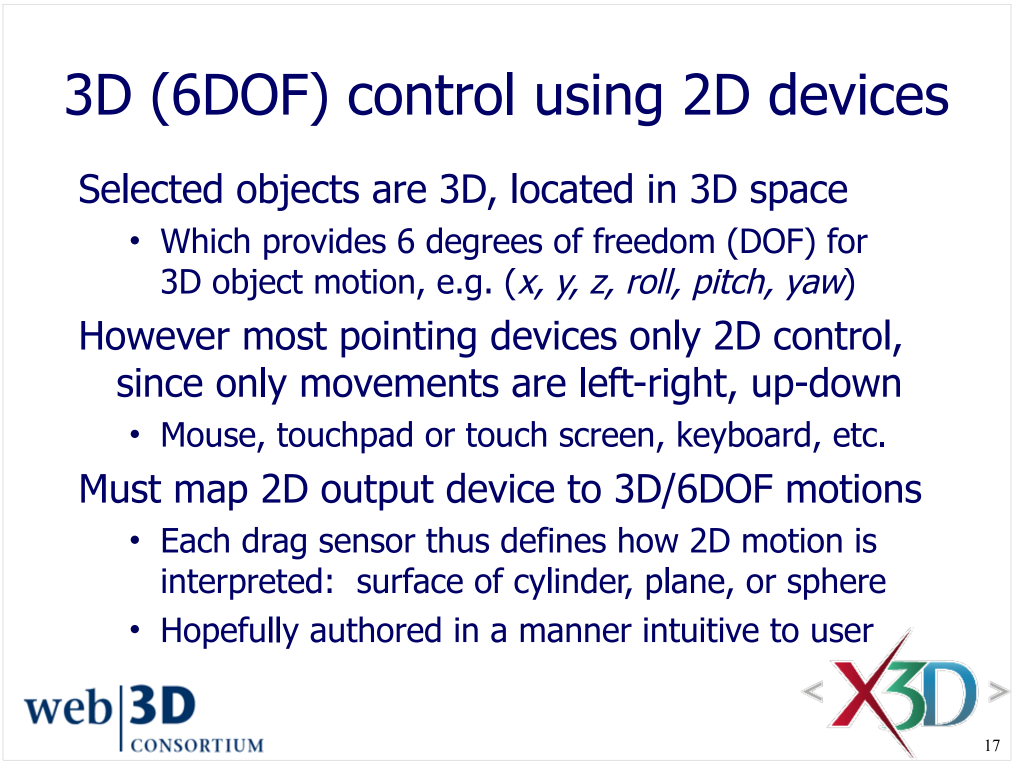
Several common fields

- *enabled, description, isActive, isOver, touchTime*

Three X3DDragSensorNode type sensors are

- CylinderSensor, PlaneSensor, SphereSensor

16



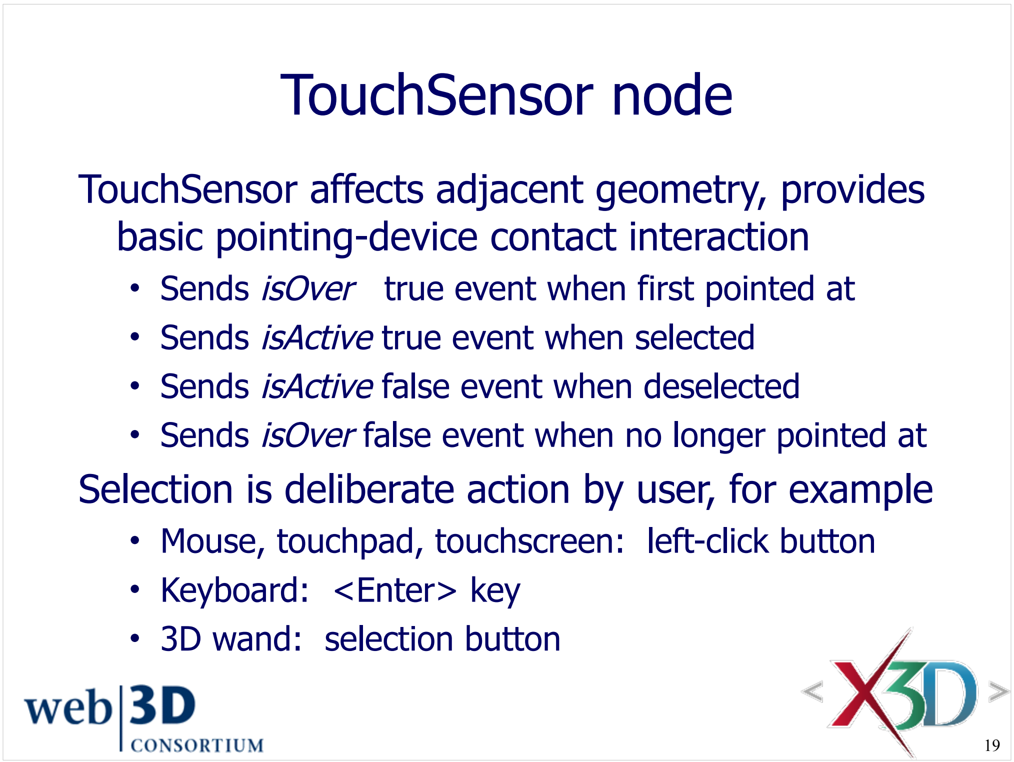
6DOF = six degrees of freedom, positional and rotational: x y z roll pitch yaw

Each of the dragging sensor nodes (CylinderSensor, PlaneSensor, SphereSensor) describe how they map 2D mouse motion (left-right, up-down) into 3D 6DOF space.

back to Table of Contents

X3D Nodes and Examples

18



A change in pointer position is needed for TouchSensor to operate.

If the geometry or camera view is animated and the geometry moves out from under the pointer, no isOver false event is sent. The pointing-device cursor icon must be moved by the user off of the selected geometry in order to send an isOver false event.

So following the initial *isOver*='true' then *isActive*='true' event pair shown on the slide, it is possible to have a slightly different order: *isOver*='false' then *isActive*='false' iff the user moves off of the selected geometry while still selected.

Sensed geometry grouping 1

All geometry that is a peer (or children of peers) of the TouchSensor nodes can be sensed

Use a grouping node (Group, Transform, etc.) to isolate sensed geometry of interest

- Don't want to make entire scene selectable, otherwise interaction isn't very sophisticated

Can attach different sensors to self-explanatory geometry for different tasks. Examples:

- Light switch *isOver* gives name, click to change
- Billboarded Text or buttons for multiple controls

web 3D CONSORTIUM

20

The Group node is an excellent way to isolate the effectiveness of a sensor to only be affected by a certain set of nodes.

Sensed geometry grouping 2

Separate sensed geometry from other shapes by using grouping nodes

Next slide shows example excerpt

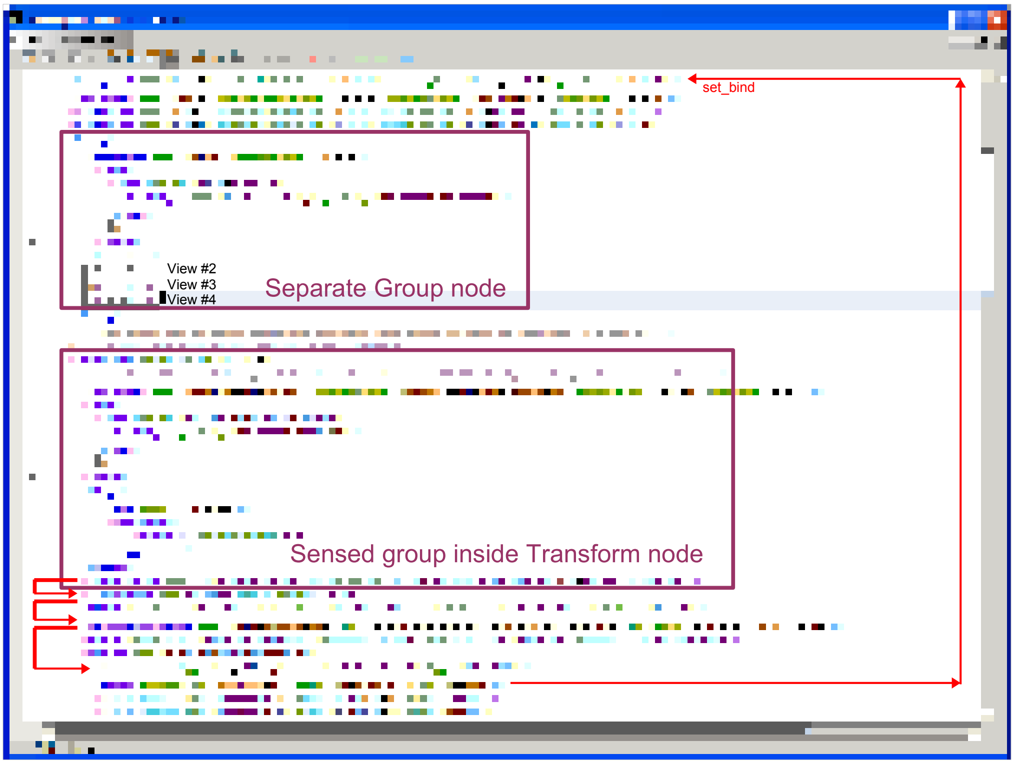
- Chapter04-ViewingNavigation/BindOperations.x3d

Scene structure for this example

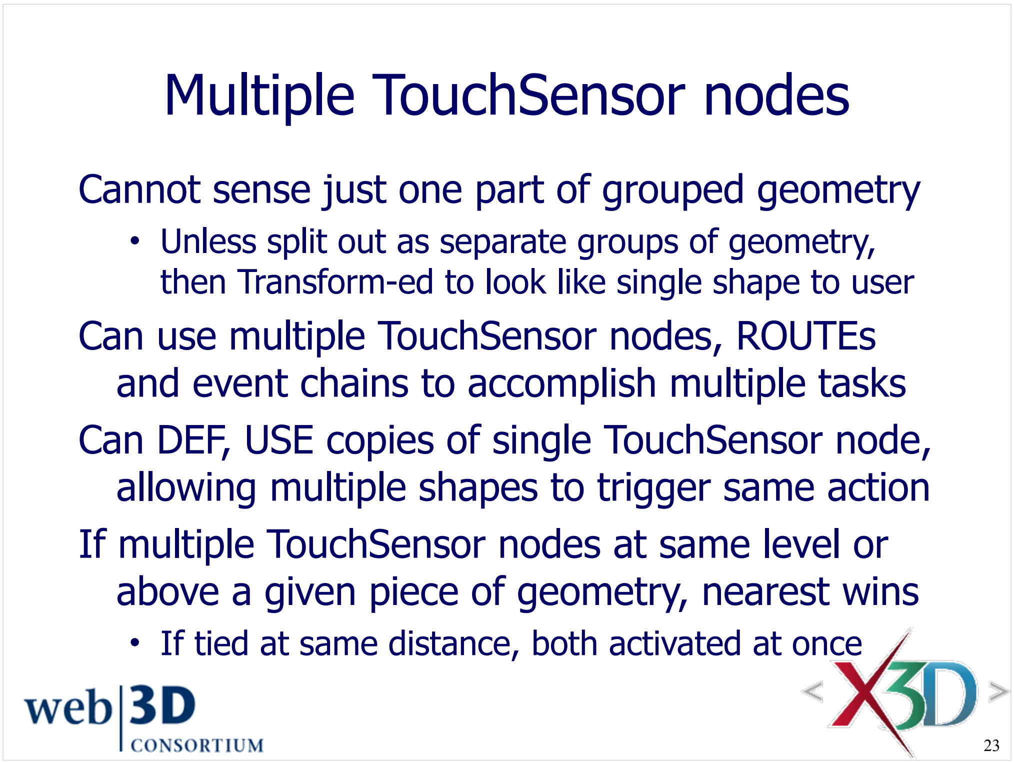
- Viewpoints consuming, producing events
- Display geometry, no sensor peer
- Selectable geometry, TouchSensor peer
- Regular animation design pattern: TimeSensor, Interpolator, target Script node, ROUTE connections

web3D CONSORTIUM

21



http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter04-ViewingNavigation/BindingOperations.x3d



Note that multiple independent TouchSensor nodes are not the same as a simultaneous multiple-touch sensor capability.

Currently X3D does not have any multi-touch (multi-hand gesture) nodes defined. Nevertheless this remains an active area of research.

- The InstantReality X3D viewer team has successfully designed and implemented multi-touch sensor capabilties. This is experimental work.
- http://instantreality.de/documentation/nodetype
- http://instantreality.de/documentation/nodetype/MultiTouchNavigator

output event *touchTime*

touchTime sends an SFTime output event whenever sensed geometry is deselected

- Sent simultaneously with *isActive* false event

Three prerequisites must be met for *touchTime:*

1. Pointing device begins pointing at sensed geometry (generating *isOver* true event)
2. Pointing device is initially activated by user selection (generating *isActive* true event)
3. Pointing device is subsequently deactivated while still pointing at the sensed geometry (generating *isActive* false event)

web3D CONSORTIUM

<X3D>

24

Because *isActive* false and *touchTime* events are sent simultaneously after meeting the same user-interaction preconditions, it is convenient to use

- *isActive* for destination fields that need boolean inputs (such as a sensor's *enabled* field)
- *touchTime* for destination fields that need SFTime inputs (such as a TimeSensor node's *set_startTime* field)

It is good design that requires a user to keep the pointer on the selected geometry before deselecting and generating a *touchTime* event. This allows a user to change their mind after initial (*isActive* true) selection, by moving the pointer off of the sensed geometry before releasing the selection. Example sequence of events:

- User selects some sensed object with pointing device
- *isActive* true event sent
- User decides that selecting the object is not desirable, and so moves the pointer off of the object before deselecting
- *isActive* false event is still sent, but no corresponding *touchTime* event is sent

As we shall see, it is possible to create event-animation logic that takes advantage of this difference.

output events *hitPoint_changed, hitNormal_changed, hitTexCoord_changed*

hitPoint_changed

- sends output SFVec3f event providing 3D location coordinates of selection point, referenced to local coordinate system

hitNormal_changed

- sends output SFVec3f event providing normal vector of underlying geometry at selection point

hitTexCoord_changed

- sends output SFVec2f event providing 2D *(u, v)* coordinates of underlying texture at selection point

web 3D CONSORTIUM

25

The local coordinate system is determined by the combined translation, rotation and scaling effects of the Transform nodes that are parent nodes for the geometry of interest. This is often referred to as the transformation hierarchy.

Normal vectors are similarly pointing in a direction that is relative to the local coordinate system.

Texture coordinates are independent of the local coordinate system, only referring to *(u,v)* coordinate values which range from 0 to 1 along each axis of a texture image. Texture coordinates are described in Chapter 5, Appearance Material and Textures.

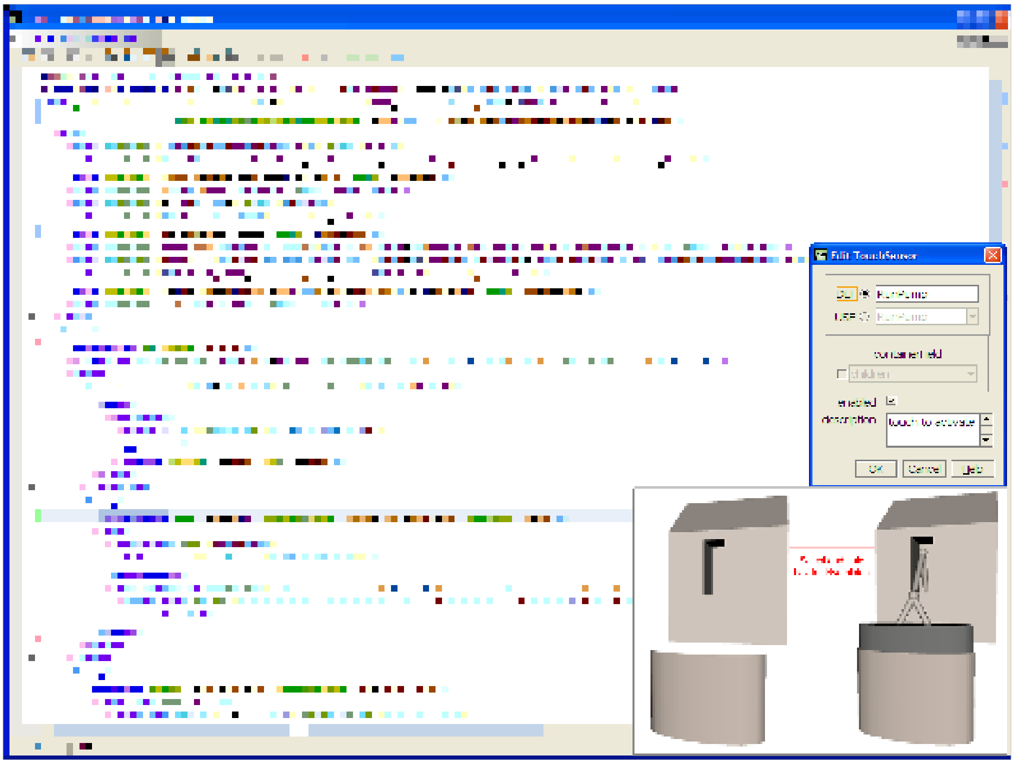


Figure 8.2, page 230, *X3D for Web Authors*

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/TouchSensorPumpHouse.x3d

Example: opening doors

Interaction in 3D scenes doesn't always have to be literal. It is easier to click on a door to open it, rather than turning a door knob.

Next example compares TouchSensor selections

- Left door opens on initial selection (click)
- Right door opens on later deselection (unclick)

Key difference: *isActive* is first true, then false

- To fix: routing events through a BooleanFilter and TimeTrigger can initiate TimeSensor appropriately
- These are Event Utility nodes, covered in Chapter 9

web 3D CONSORTIUM

27

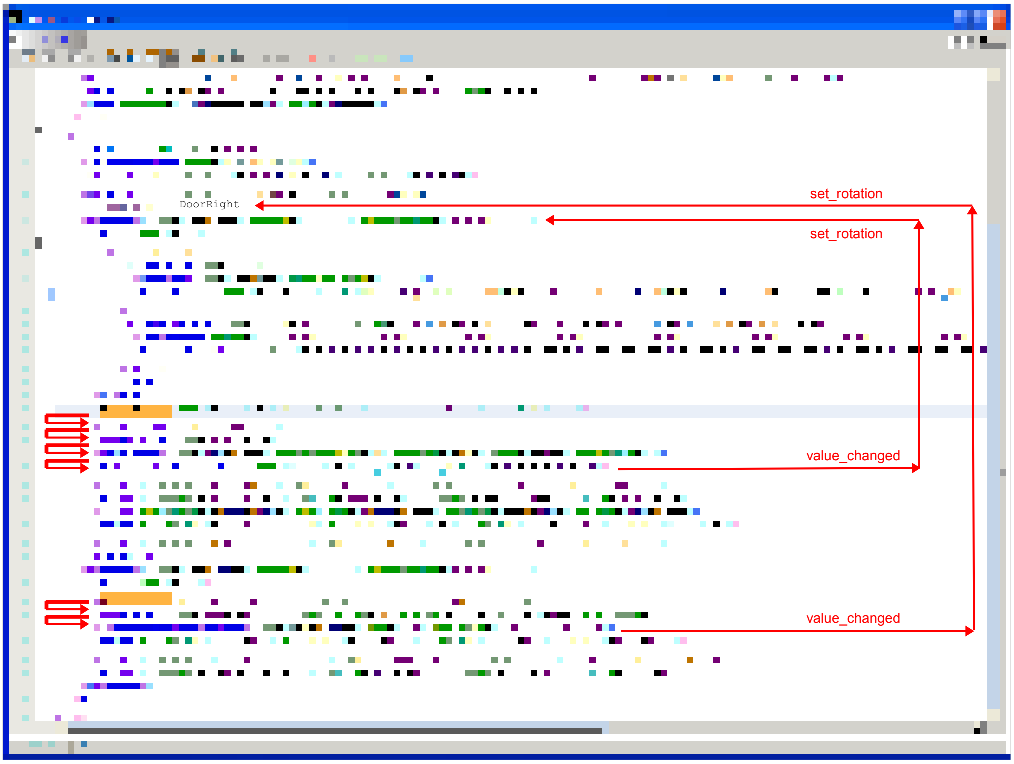
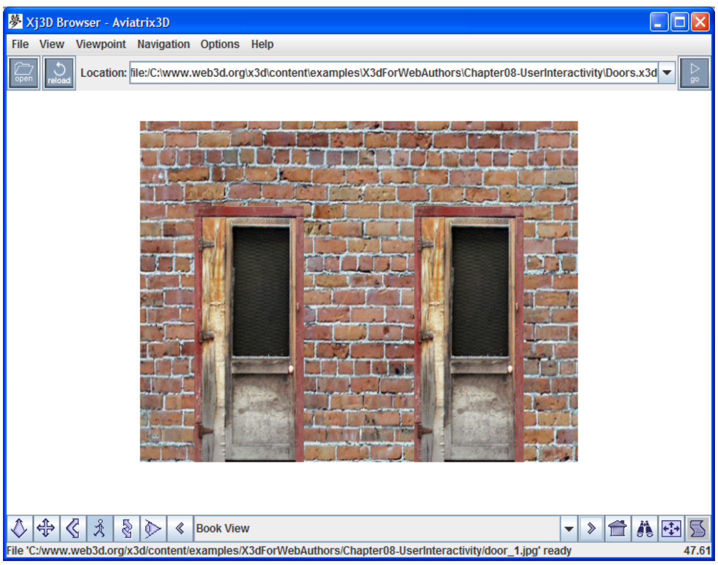
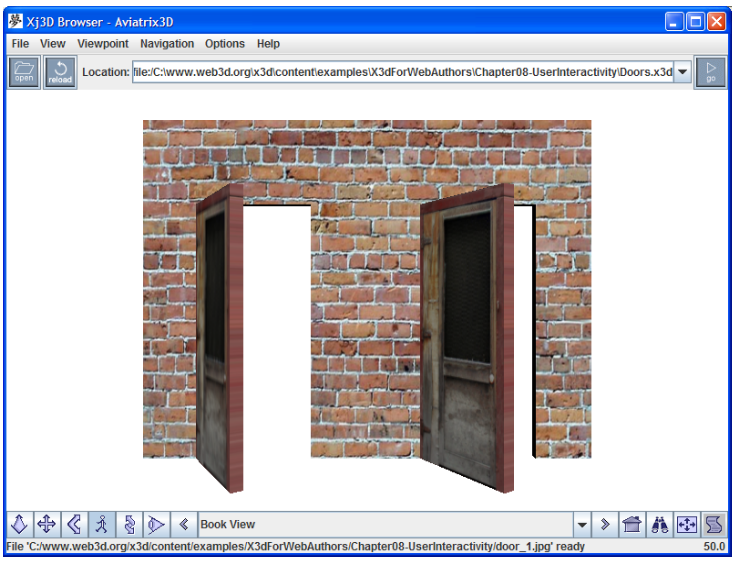


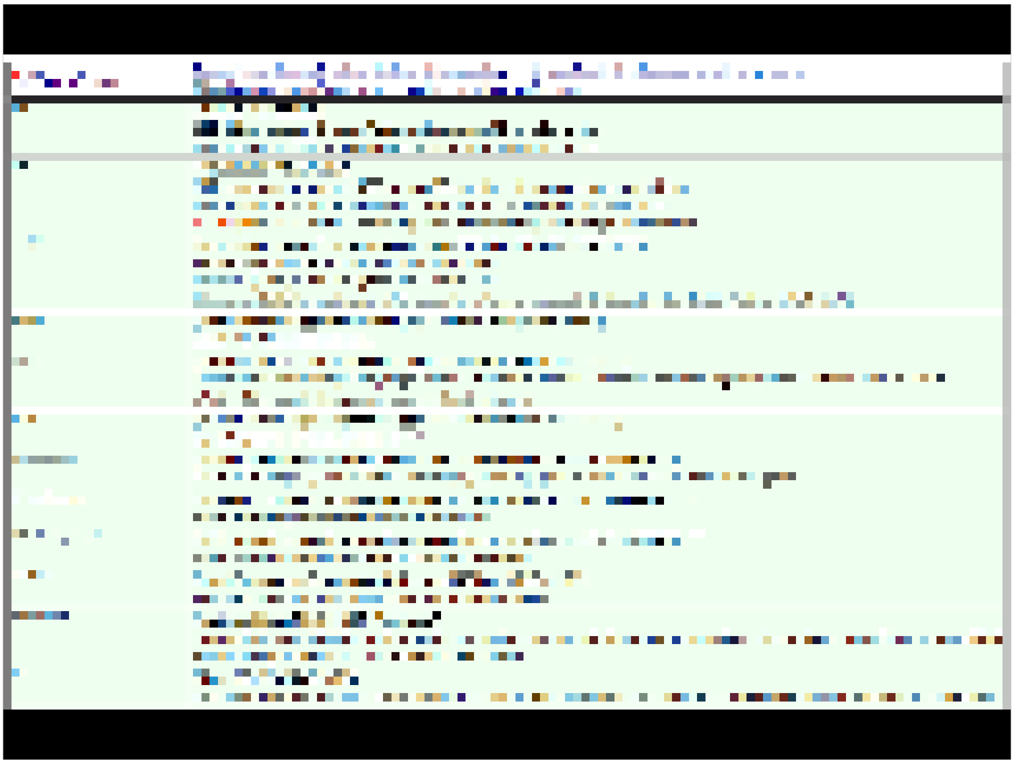
Figure 8.1, page 229, *X3D for Web Authors*

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/Doors.x3d

The following snapshots show animation results after clicking on (selecting) each door.







http://www.web3d.org/x3d/content/X3dTooltips.html#TouchSensor

PlaneSensor node

PlaneSensor converts x-y dragging motion by the pointing device into lateral translation in plane

- 2-tuple motion converted to 3-tuple SFVec3f
- Motion is parallel to local z=0 plane (screen plane)

Activated by peer geometry in scene graph

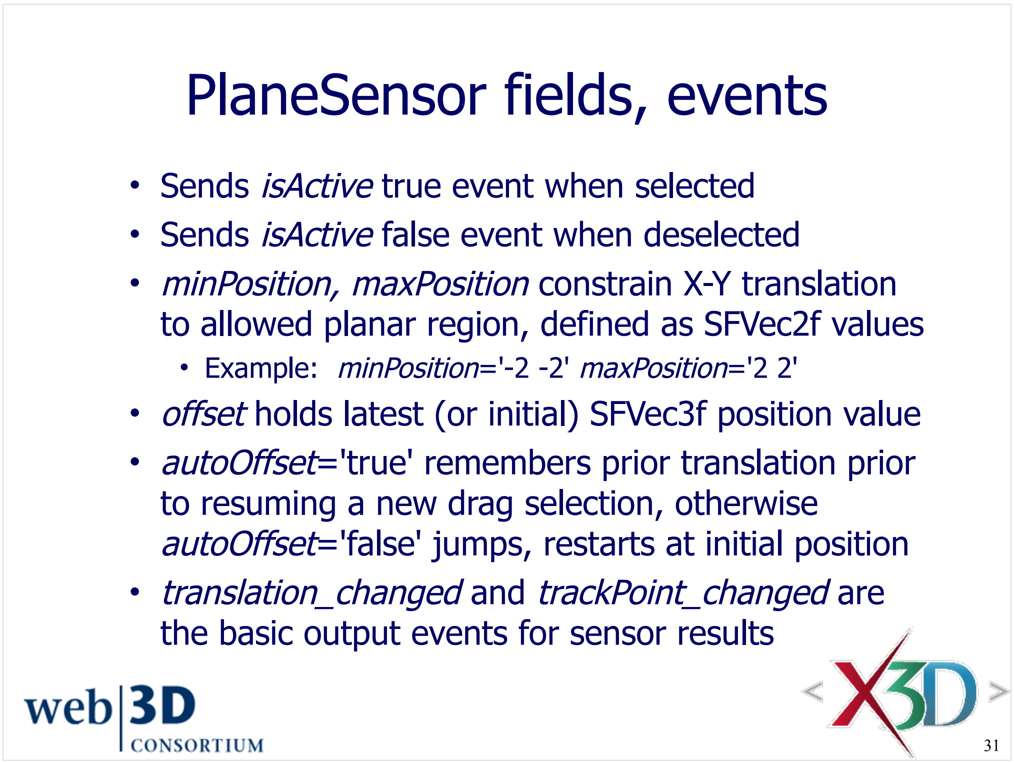
- Sensor itself is not rendered, unless background geometry or sensed shape itself has a planar side

Translation output values can follow a ROUTE connection to parent Transform *translation*

- Or connect to another SFVec3f field elsewhere

web3D CONSORTIUM

<X3D>



Default values *minPosition*='0 0' *maxPosition*='-1 -1' are contradictory (minimum values are greater than corresponding maximum values), which results in the PlaneSensor being unconstrained.

These constraints are helpful for guiding the user to make reasonable adjustments, rather than dragging something off into the far distance somewhere.

Pay close attention to user viewpoint and perspective across the full range of possible movement, so that dragged geometry remains visible and accessible for further adjustment.

Linear movement can be achieved by setting either the min/max X or else min/max Z constraints to the same value. This is done in the next example, PlaneSensorPumpHouse.x3d.

As with all sensors, PlaneSensor includes *description* and *enabled* fields.

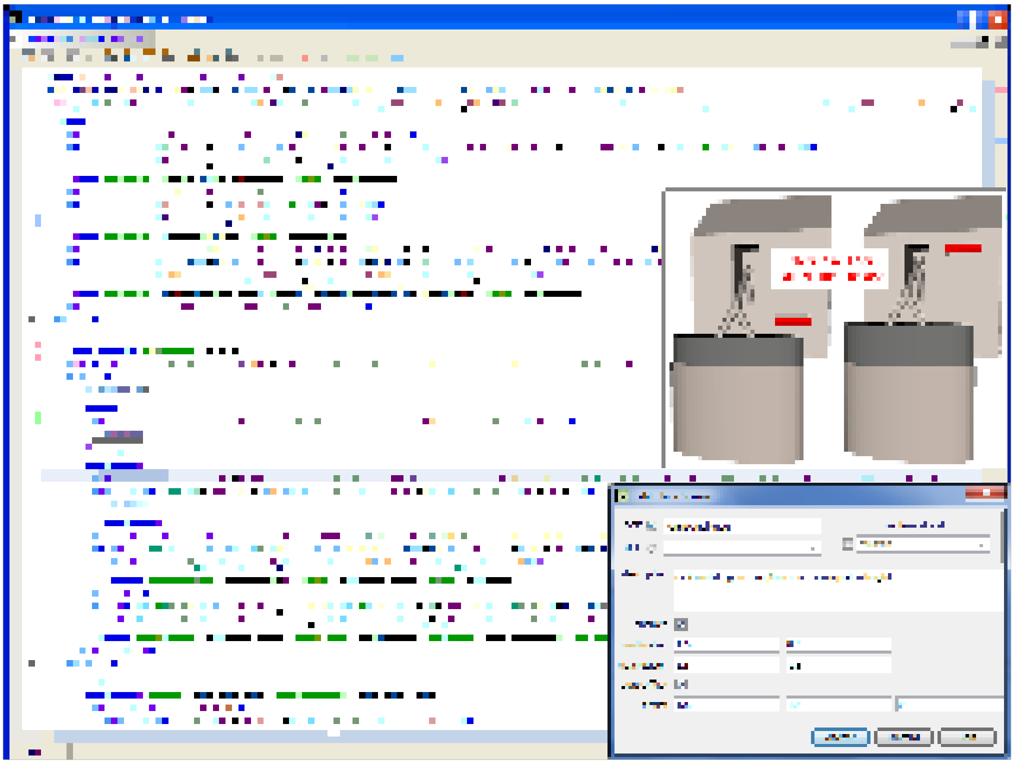
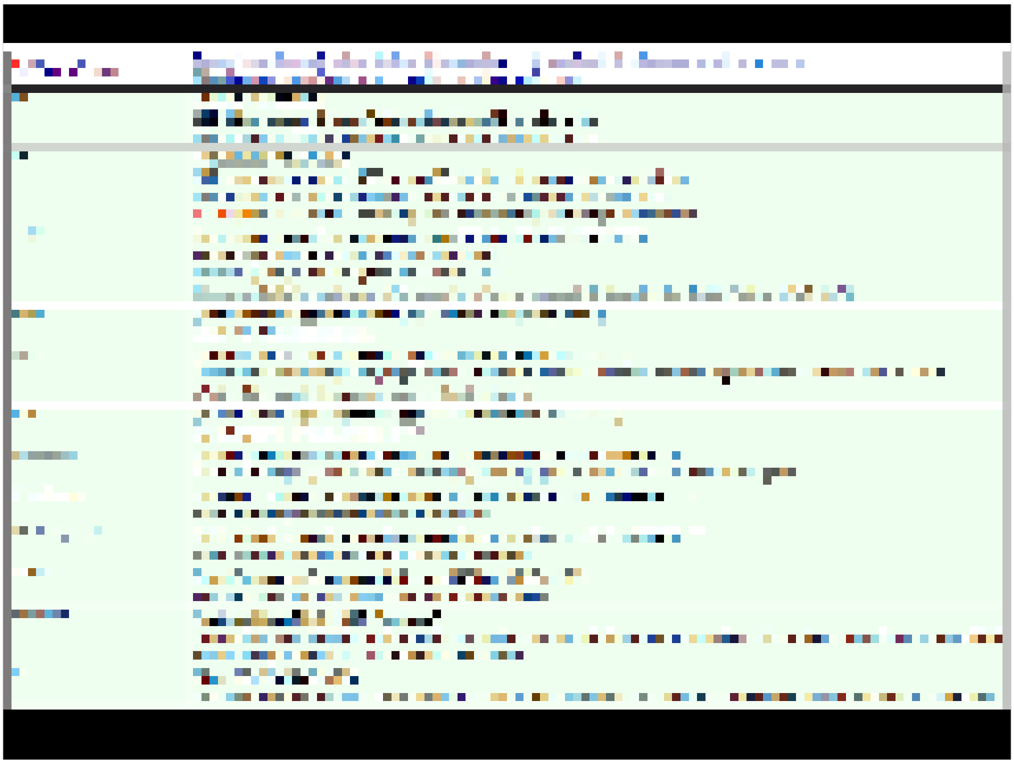


Figure 8.3, page 233, X3D for Web Authors

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/PlaneSensorPumpHouse.x3d



http://www.web3d.org/x3d/content/X3dTooltips.html#PlaneSensor

CylinderSensor node

CylinderSensor converts x-y dragging motion by the pointing device into rotation about an axis

- 2-tuple motion converted to 4-tuple SFRotation
- Rotation restricted to local coordinate frame y-axis

Activated by peer geometry in scene graph

- Sensor itself is not rendered, unless sensed shape is itself cylindrical

Rotation output values can follow a ROUTE connection to parent Transform *rotation*

- Or connect to another SFRotation field elsewhere

web3D CONSORTIUM

X3D

34

PlaneSensor gets from X-Y values to X-Y-Z values by simply holding Z to equal 0.

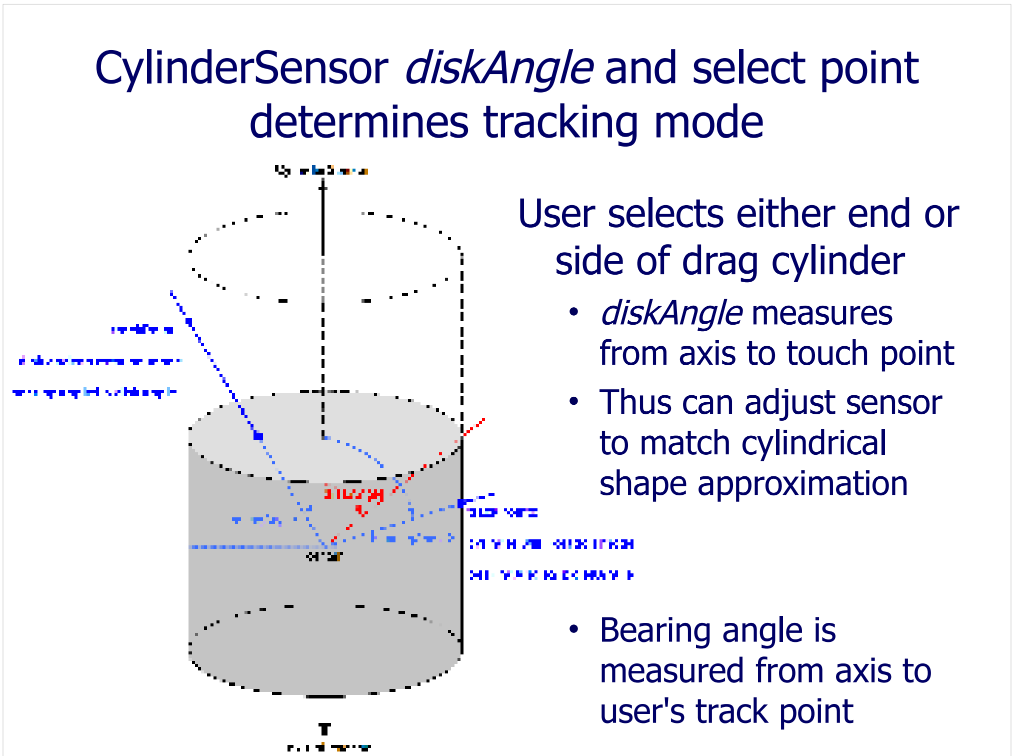


Figure 8.4, p. 236, *X3D for Web Authors*

User selection with pointing device defines the track point. The vector angle of the track point relative to the *diskAngle* parameter determines whether CylinderSensor responds in end-cap tracking mode, or cylinder-wall tracking mode.

Each mode has a slightly different way of responding to user dragging motions, making response more intuitive if there is a good match to the geometry.

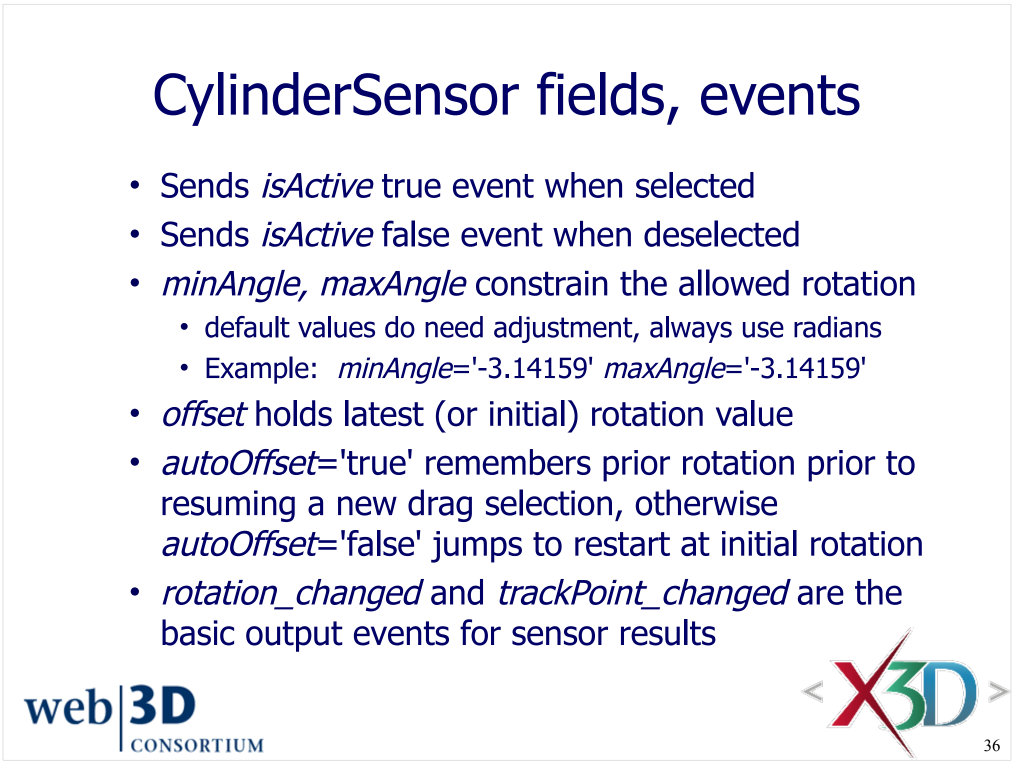
CylinderSensor can be forced to always operate in end-cap mode by setting *diskAngle*='1.5707' (π/2 radians), which is useful to emulate turning of knobs.

CylinderSensor can be forced to always operate in cylinder-wall sides mode by setting *diskAngle*='0' which is useful to emulate a thumbwheel rotation.

Angle relationships are measured as SFFloat radian values, while node output field *rotation_changed* is SFRotation.

The cylinder shown in the above diagram is typically invisible and describes the mathematical model of the sensor response. If the actual sensed geometry is not particularly cylindrical in shape, sometimes superimposing a semitransparent cylinder can make the reaction more obvious. For example prototypes, see

https://savage.nps.edu/Savage/Tools/Animation/ArbitraryAxisCylinderSensorExamples.x3d



As with all sensors, CylinderSensor includes *description* and *enabled* fields.

CylinderSensor off-axis rotation design pattern 1

- CylinderSensor rotates about the Y axis of local coordinate frame
  - No internal field provided for offsetting that axis
  - Making rotation axis different than peer sensed geometry can be tricky
- Following scene-graph design pattern shows how to rotate CylinderSensor about a different axis
  - First rotate to desired axis, CylinderSensor is child
  - Nest a second Transform rotation restoring original Y-axis, place sensed geometry here as child

web3D CONSORTIUM

37

This pattern works because a Sensor node acts upon all of its peers, and all of its peers' children.

Essentially both CylinderSensor and geometry are rotated to the new angle of interest, than the geometry is rotated back to its original orientation.

CylinderSensor off-axis rotation design pattern 2

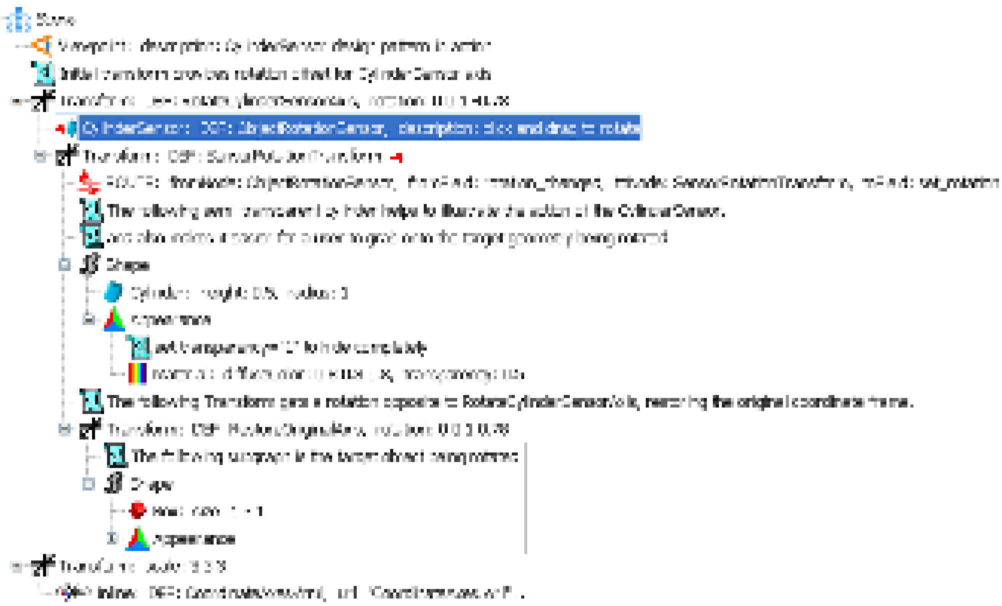


Figure 8.6, p. 238, *X3D for Web Authors*

Note that the CylinderSensor is nested within the animated Transform, making the rotation changes an interesting feedback loop.

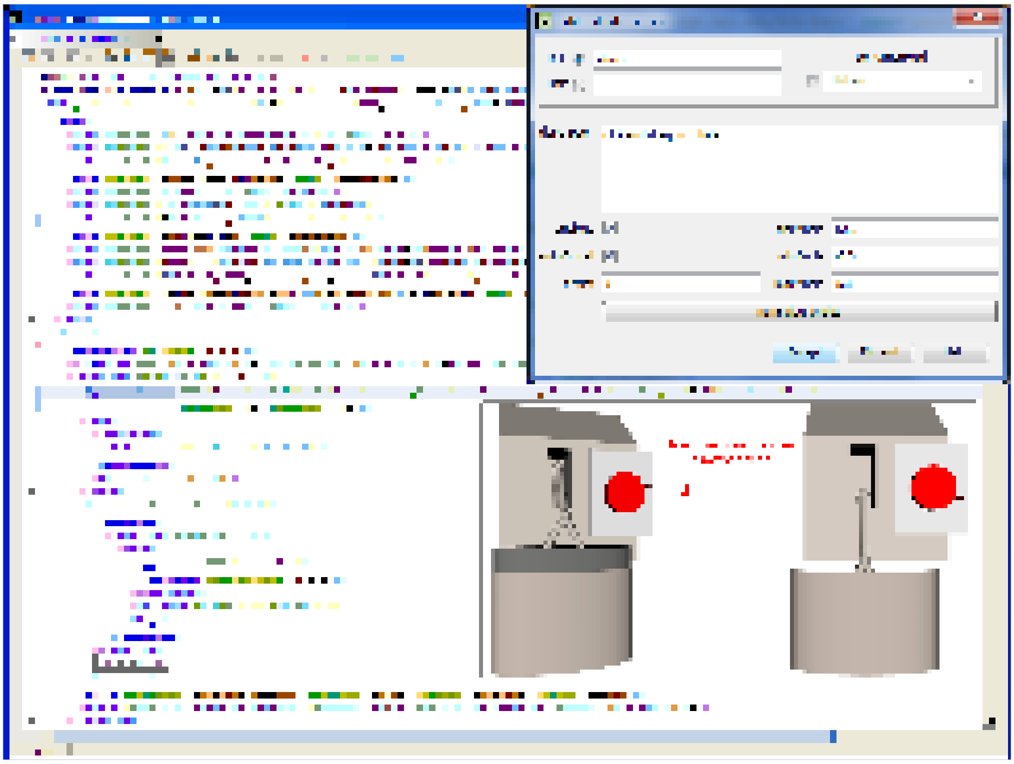
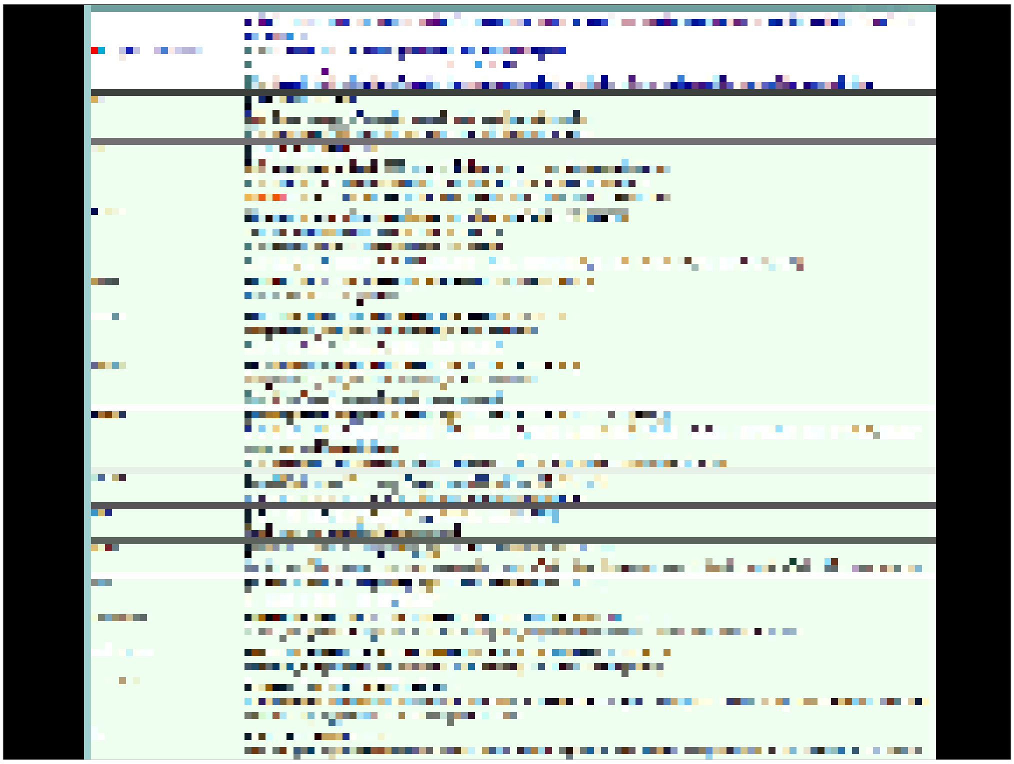


Figure 8.5, p. 237, *X3D for Web Authors*

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/CylinderSensorPumpHouse.x3d



http://www.web3d.org/x3d/content/X3dTooltips.html#CylinderSensor

SphereSensor node

SphereSensor converts x-y dragging motion by the pointing device into an arbitration rotation

- 2-tuple motion converted to 4-tuple SFRotation
- Rotation about origin of local coordinate frame

Activated by peer geometry in scene graph

- Sensor itself is not rendered, unless corresponding sensed shape itself happens to be spherical

Rotation output values can have ROUTE connection to parent Transform *rotation* field

- Or connected to another SFRotation field elsewhere

web 3D CONSORTIUM

<X3D>

41

SphereSensor fields, events

- Sends *isActive* true event when selected
- Sends *isActive* false event when deselected
- *offset* holds latest (or initial) rotation value
- *autoOffset*='true' remembers prior rotation prior to resuming a new drag selection, otherwise *autoOffset*='false' jumps to restart at initial rotation
- *rotation_changed* and *trackPoint_changed* are the basic output events for sensor results

As with all sensors, includes *description*, *enabled*

web|3D CONSORTIUM

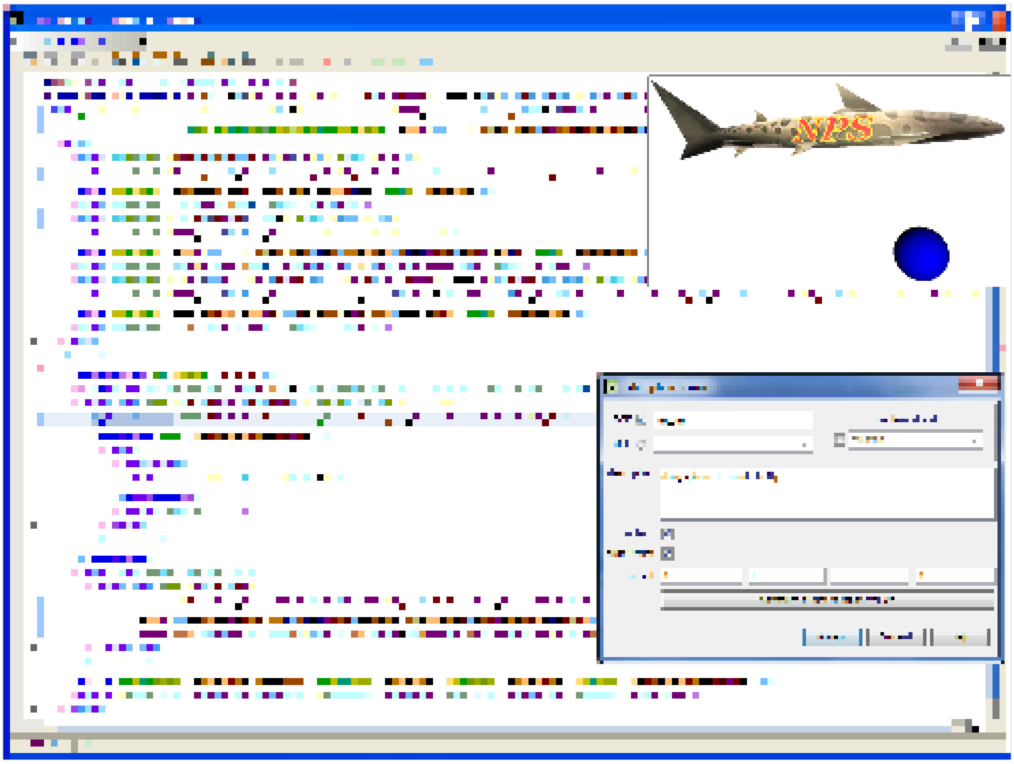
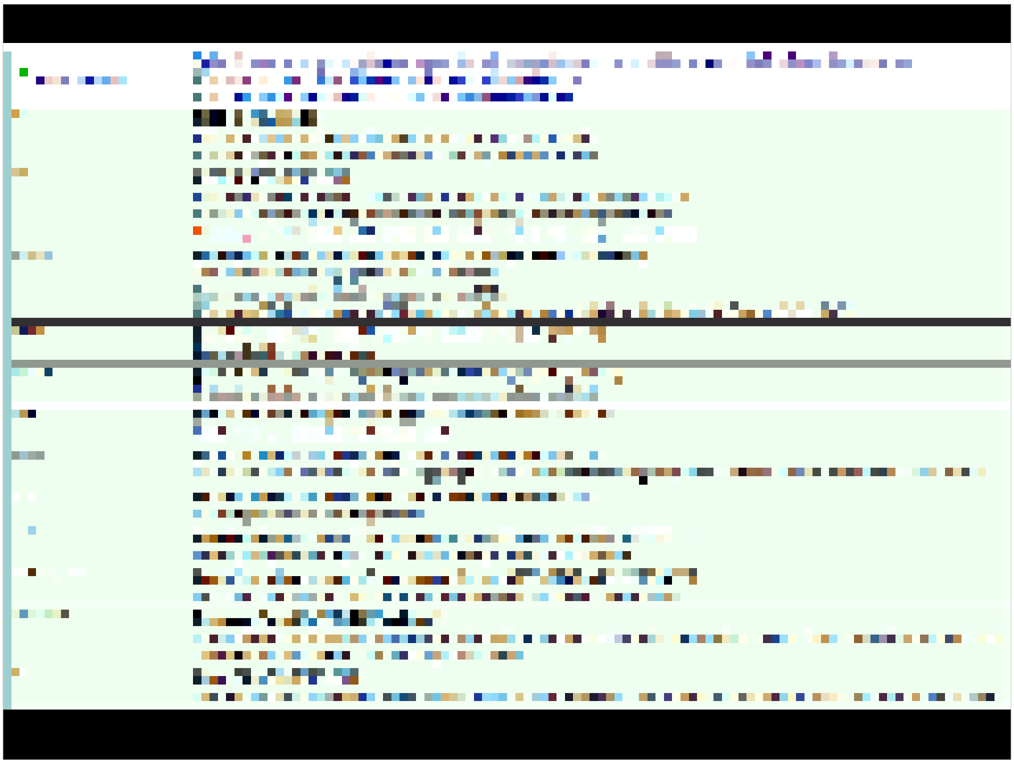


Figure 8.7, p. 241, *X3D for Web Authors*

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/SphereSensor-Lefty.x3d



http://www.web3d.org/x3d/content/X3dTooltips.html#SphereSensor

KeySensor node

KeySensor is a one-character-at-a-time interface, capturing key presses from user's keyboard
- Helpful for selecting from menu choices
- Helpful for creating a special keyboard-driven navigation interface
- Only gives key name, not precise shifted character

Control, alt, shift keys sent as separate events
- As are certain special "action keys"

Processing key events requires a Script node
- Covered in Chapter 9, Event Utilities and Scripting

web 3D CONSORTIUM

45

KeySensor events 1

- Sends *isActive* true event when selected
- Sends *isActive* false event when deselected
- *keyPress*, *keyRelease* provide SFString value for the specific key pressed (or released)
  - Usually upper-case or primary key symbol only
- *shiftKey, altKey, controlKey* are SFBool binary values indicating whether keys were pressed or released

KeySensor also has *enabled* field

- but not *description* since display is challenging

web 3D CONSORTIUM

46

KeySensor includes *description* and *enabled* fields.

KeySensor events 2

- *actionKeyPress, actionKeyRelease* provide SFInt32 values when pressed or released

Key	Value	Interaction Default
F1-F12	1-12	
Home	13	First viewpoint
End	14	Last viewpoint
PageUp	15	Prior viewpoint
PageDown	16	Next viewpoint
Arrow up	17	Cursor up
Arrow down	18	Cursor down
Arrow left	19	Cursor left
Arrow right	20	Cursor right

Table 8.15, page 243, *X3D for Web Authors*

Be careful not to unintentionally override default navigation behaviors for above keys.

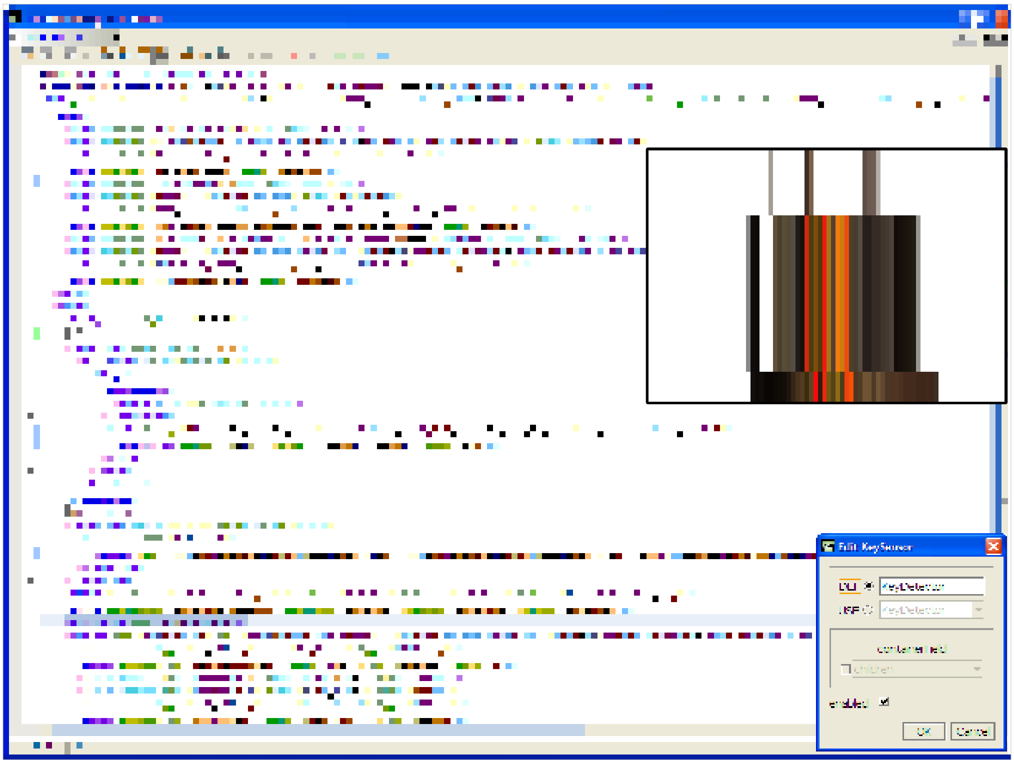
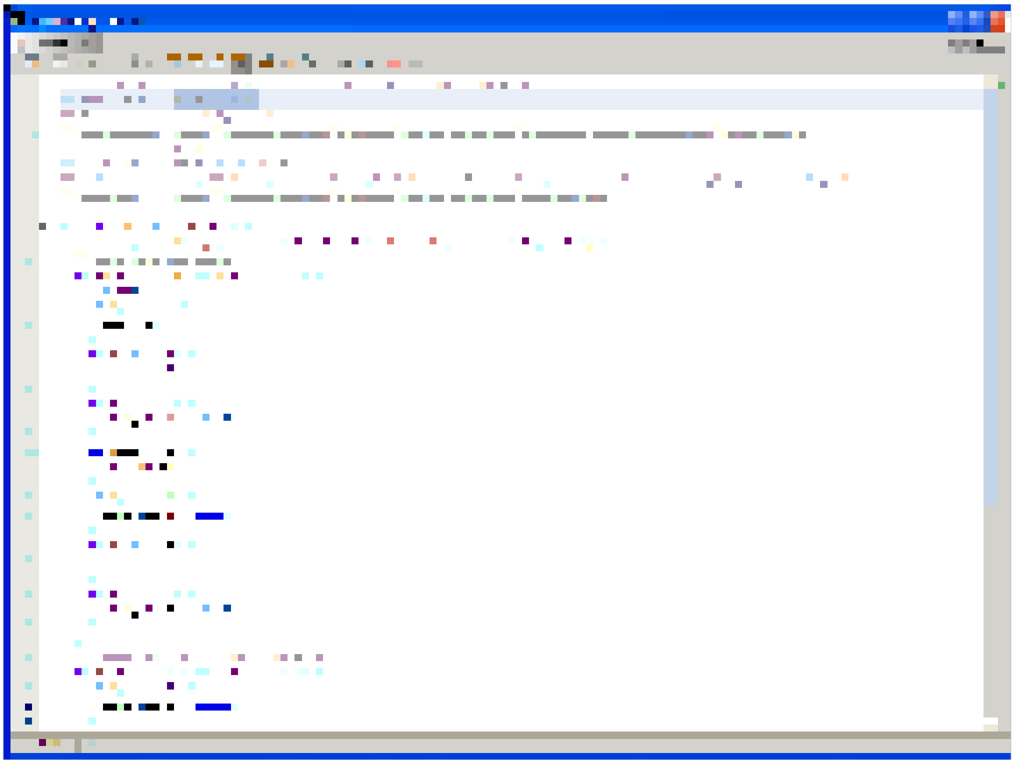


Figure 8.8, page 244, *X3D for Web Authors*

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/KeySensor-Lefty.x3d



keySensor.js is invoked by a Script node in KeySensor-Lefty.x3d in order to process user key presses and output viewpoint binding events.

Script nodes are covered in Chapter 9, Event Utilities and Scripting.

http://www.web3d.org/x3d/content/X3dTooltips.html#KeySensor

StringSensor node, events

StringSensor provides a string-based interface to the user's keyboard

- Each character key press is collected until <Enter> key is returned, completing *finalText* string
- Intermediate string results (including deletions) also available as user proceeds in *enteredText* string
- *deletionAllowed* is boolean field that enables <Backspace>, <Delete> keys

StringSensor has *isActive* events, *enabled* field

- but not *description* since display is challenging

web3D CONSORTIUM

<X3D>

51

If displaying entered text, you may want to provide a colored Box background behind it in order to improve contrast and readability without clutter from the surrounding scene.

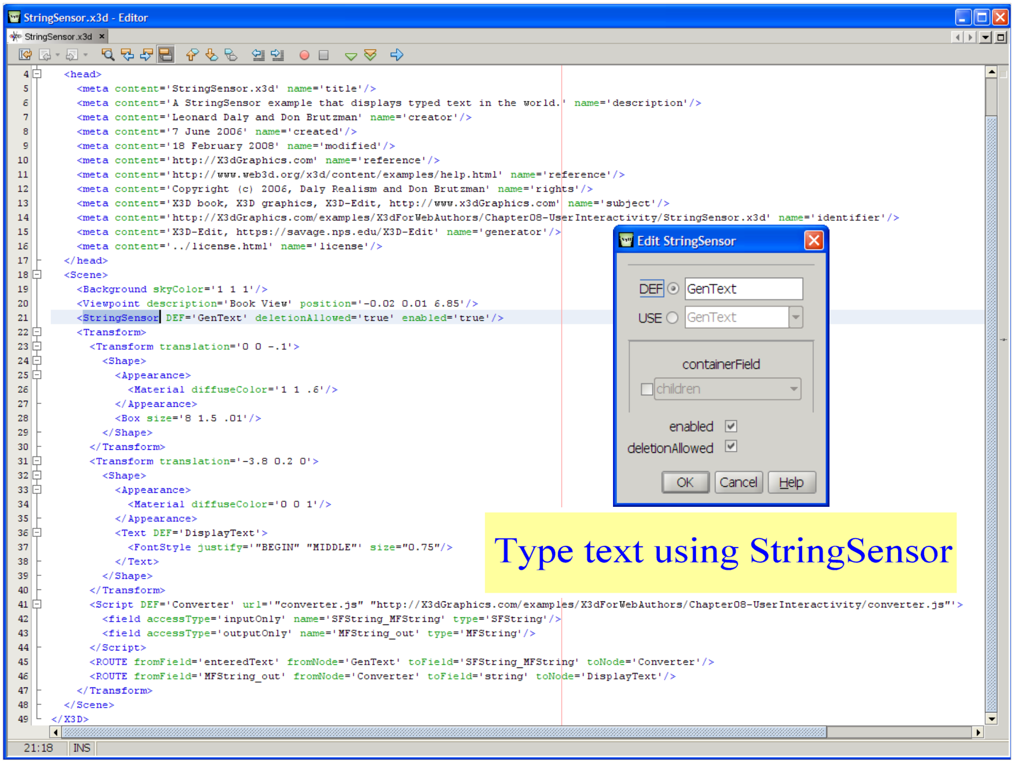
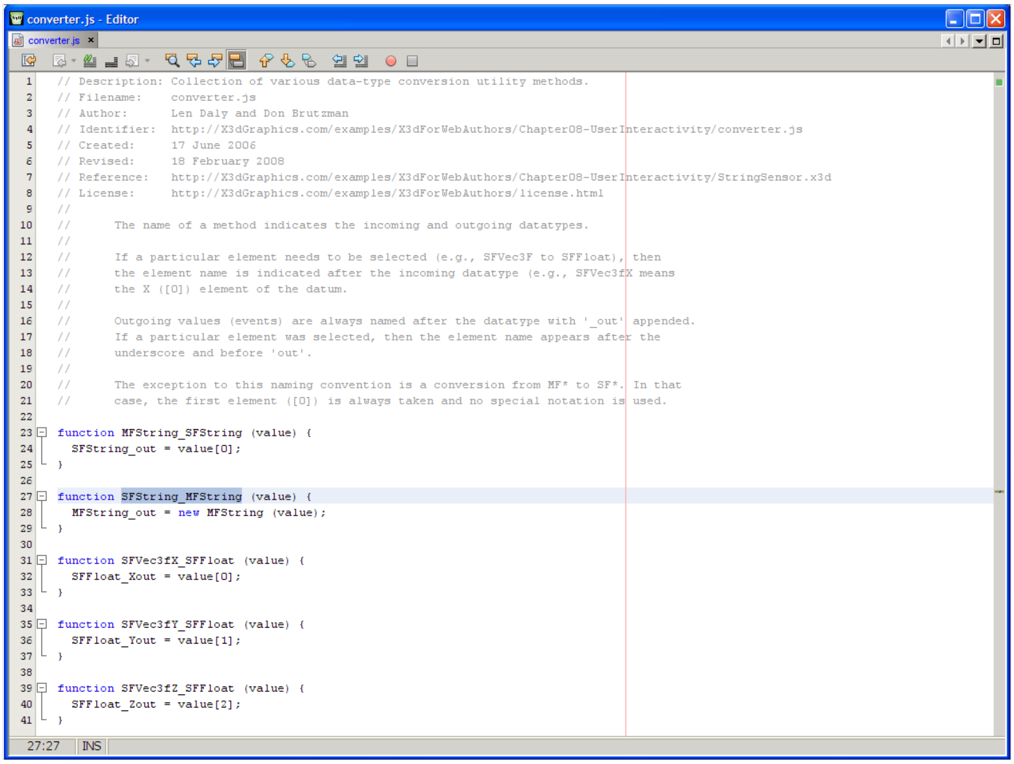


Figure 8.9, page 246, *X3D for Web Authors*

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/StringSensor.x3d



converter.js is invoked by a Script node in StringSensor.x3d in order to use the following type-conversion function:

```
function SFString_MFString (value) {
    MFString_out = new MFString (value);
}
```

This script is necessary to convert the SFString output of the StringSensor node *enteredText* field into the MFString input needed for the Text node *string* field. In other words, a single SFString is converted into a MFString array with a single element.

Script nodes are covered in Chapter 9, Event Utilities and Scripting.

StringSensor	StringSensor generates events as the user presses keys on the keyboard.
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. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute!
enabled	**[enabled: accessType inputOutput, type SFBool (true|false) "true"]** Enables/disables node operation.
deletionAllowed	**[deletionAllowed: accessType inputOutput, type SFBool (true|false) "true"]** If deletionAllowed is true, then previously entered character in enteredText can be removed. If deletionAllowed is false, then characters may only be added to the string. Hint: deletion key is typically defined by local system.
isActive	**[isActive: accessType outputOnly, type SFBool (true|false) #FIXED ""]** isActive true/false events are sent when triggering the sensor. isActive=true when primary mouse button is pressed, isActive=false when released.
enteredText	**[enteredText: accessType outputOnly, type SFString CDATA #FIXED ""]** Events generated as character-producing keys are pressed on keyboard.
finalText	**[finalText: accessType outputOnly, type SFString CDATA #FIXED ""]** Events generated when sequence of keystrokes matches keys in terminationText string when this condition occurs, enteredText is moved to finalText and enteredText is set to empty string. Hint: termination key is typically defined by local system.
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.

http://www.web3d.org/x3d/content/X3dTooltips.html#StringSensor

Example: user-interactivity sensor nodes

UserInteractivitySensorNodes.x3d

- Select (click and hold) TouchSensor Cone to alternate Background nodes
- Select and drag PlaneSensor -- Box on the screen
- Select and drag to rotate CylinderSensor -- Cylinder
- Select and drag to spin SphereSensor -- Sphere

Keyboard inputs are also activated

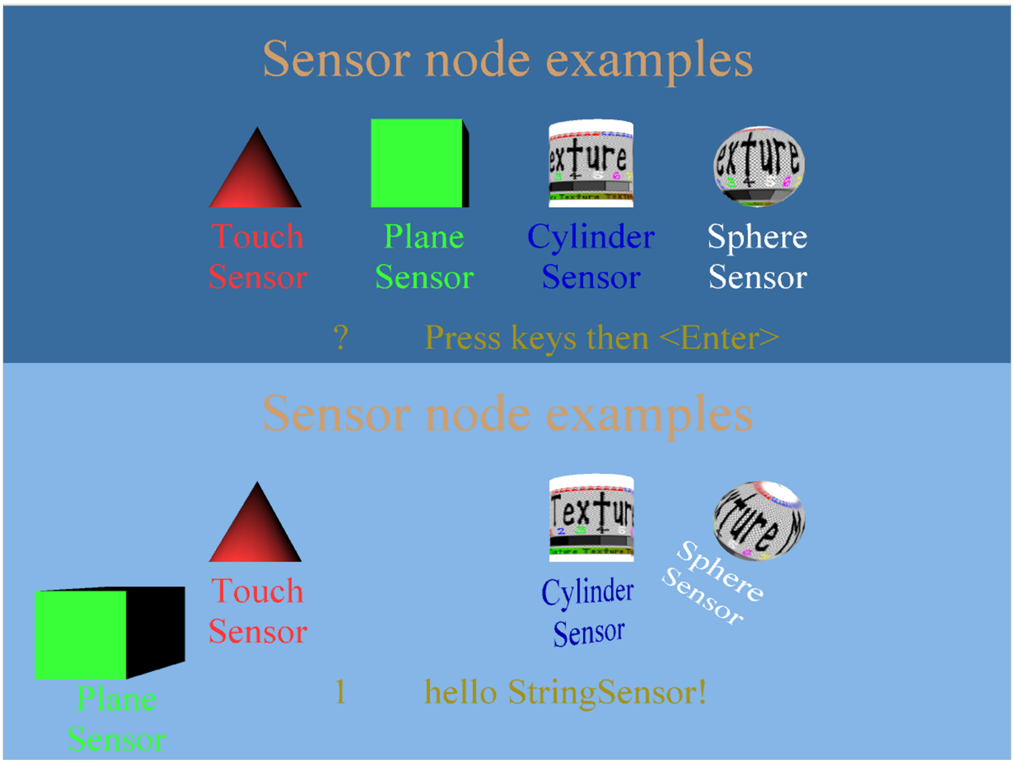
- KeySensor indicates keyPress
- StringSensor shows *finalText* once <Enter> pressed
- Console shows *enteredText* (includes deletes if any)

web3D CONSORTIUM

X3D

55

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/UserInteractivitySensorNodes.x3d



http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/UserInteractivitySensorNodes.x3d

The top screen is the initial view. Click and hold to select the Cone TouchSensor that binds the light-blue Background. Releasing unbinds that Background, restoring the original.

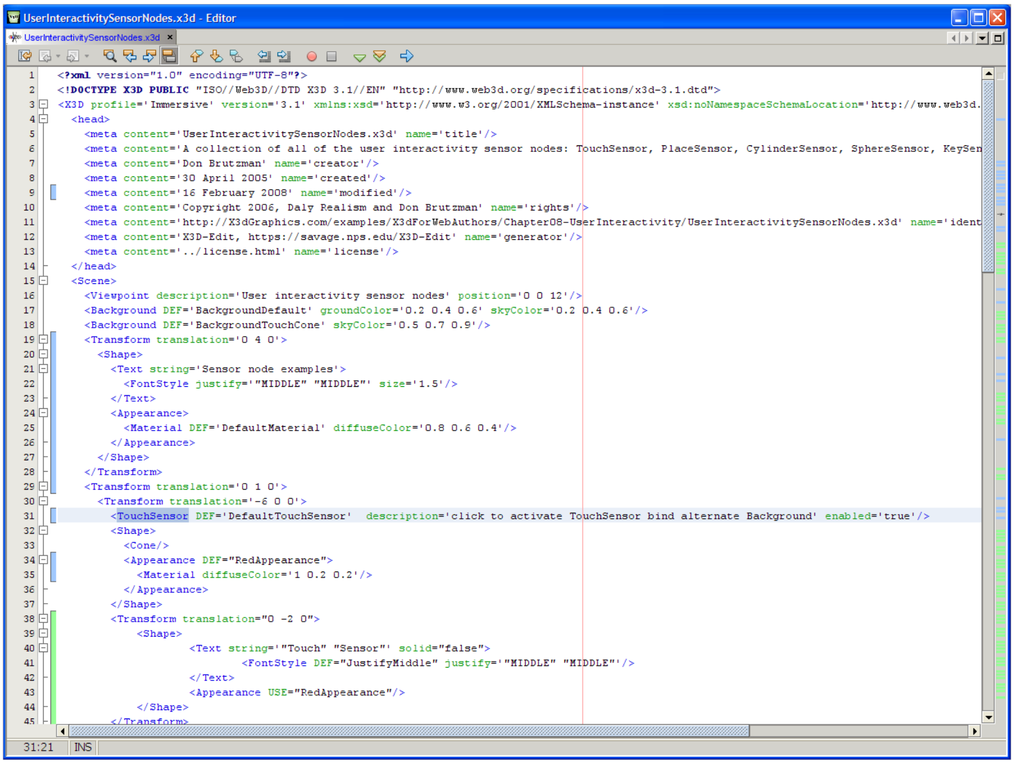
PlaneSensor, CylinderSensor and SphereSensor can each be selected and dragged. Their output values (SFVec3f, SFRotation, SFRotation) have ROUTE connections to either translate or rotate the respective parent Transform node.

Default KeySensor output text is a ? question mark. Note that the key output shows only a capital-letter character (or the primary character) for the key being pressed.

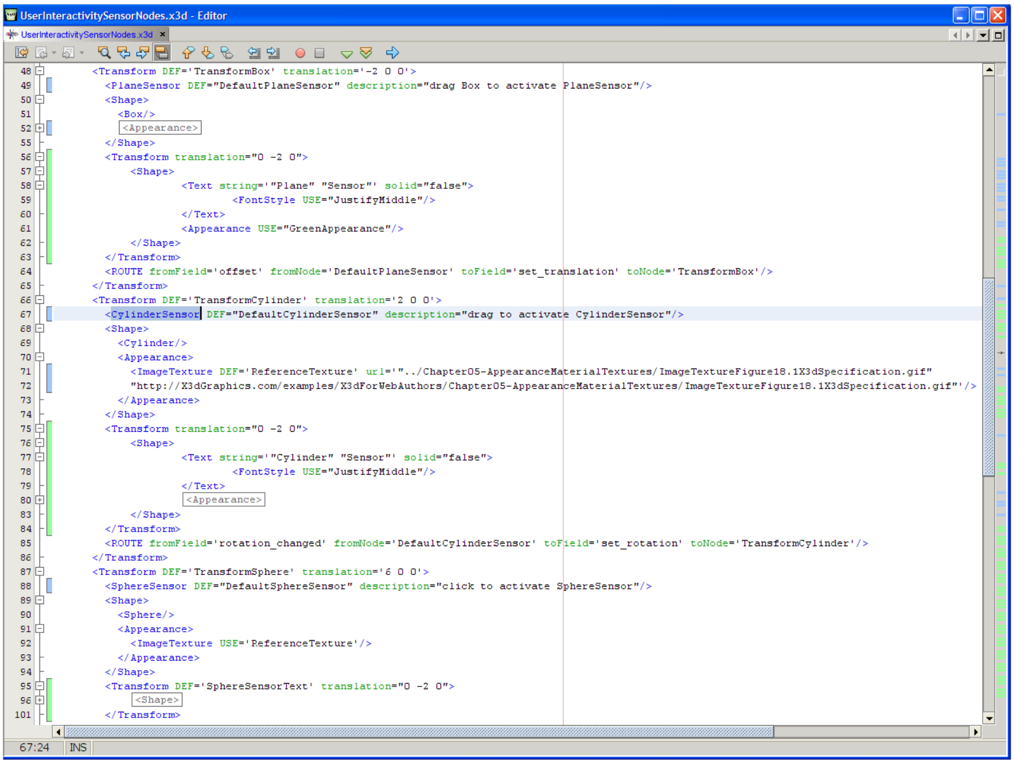
Default StringSensor output text is 'Press keys then <Enter>' - be patient since the *finalText* field doesn't send an output string until the <Enter> key is pressed.

The console shows the *enteredText*, as it is typed key by key, including <Backspace> or <Delete> effects (if any).

```
enteredText=H
enteredText=He
enteredText=Hel
enteredText=Hell
enteredText=Hello
enteredText=Hello
enteredText=Hello S
enteredText=Hello St
enteredText=Hello Str
enteredText=Hello Stri
enteredText=Hello Strin
enteredText=Hello String
enteredText=Hello StringS
enteredText=Hello StringSe
enteredText=Hello StringSen
enteredText=Hello StringSens
enteredText=Hello StringSenso
enteredText=Hello StringSensor
enteredText=Hello StringSensor!
enteredText=Hello StringSensor!
```



http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/UserInteractivitySensorNodes.x3d




```
<Transform DEF='TransformBox' translation='-2 0 0'>
  <PlaneSensor DEF="DefaultPlaneSensor" description="drag Box to activate PlaneSensor"/>
  <Shape>
    <Box/>
    <Appearance>
  </Shape>
  <Transform translation="0 -2 0">
      <Shape>
              <Text string='"Plane" "Sensor"' solid="false">
                      <FontStyle USE="JustifyMiddle"/>
              </Text>
              <Appearance USE="GreenAppearance"/>
      </Shape>
  </Transform>
  <ROUTE fromField='offset' fromNode='DefaultPlaneSensor' toField='set_translation' toNode='TransformBox'/>
</Transform>
<Transform DEF='TransformCylinder' translation='2 0 0'>
  <CylinderSensor DEF="DefaultCylinderSensor" description="drag to activate CylinderSensor"/>
  <Shape>
    <Cylinder/>
    <Appearance>
      <ImageTexture DEF='ReferenceTexture' url='"../Chapter05-AppearanceMaterialTextures/ImageTextureFigure18.1X3dSpecification.gif"
      "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter05-AppearanceMaterialTextures/ImageTextureFigure18.1X3dSpecification.gif"'/>
    </Appearance>
  </Shape>
  <Transform translation="0 -2 0">
      <Shape>
              <Text string='"Cylinder" "Sensor"' solid="false">
                      <FontStyle USE="JustifyMiddle"/>
              </Text>
              <Appearance>
      </Shape>
  </Transform>
  <ROUTE fromField='rotation_changed' fromNode='DefaultCylinderSensor' toField='set_rotation' toNode='TransformCylinder'/>
</Transform>
<Transform DEF='TransformSphere' translation='6 0 0'>
  <SphereSensor DEF="DefaultSphereSensor" description="click to activate SphereSensor"/>
  <Shape>
    <Sphere/>
    <Appearance>
      <ImageTexture USE='ReferenceTexture'/>
    </Appearance>
  </Shape>
  <Transform DEF='SphereSensorText' translation="0 -2 0">
      <Shape>
  </Transform>
```

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/UserInteractivitySensorNodes.x3d

UserInteractivitySensorNodes.x3d - Editor

```
                <Shape>
                        <Text DEF='KeyText' solid="false" string='?'>
                                <FontStyle USE="JustifyMiddle"/>
                        </Text> <Appearance>
                </Shape>
            </Transform>
            <Transform translation="-2 -3 0">
                <Shape>
                        <Text DEF='StringText' solid="false" string='Press keys then &lt;Enter&gt;'>
                                <FontStyle justify='"BEGIN" "MIDDLE"'/>
                        </Text> <Appearance USE="BrownAppearance"/>
                </Shape>
            </Transform>
        <KeySensor DEF='DefaultKeySensor'  enabled='true'/>
        <StringSensor DEF='DefaultStringSensor'  deletionAllowed='true' enabled='true'/>
        <Script DEF="KeyboardProcessor">
            <field name='keyInput' type='SFString' accessType='inputOnly'/>
            <field name='finalTextInput' type='SFString' accessType='inputOnly'/>
            <field name='enteredTextInput' type='SFString' accessType='inputOnly'/>
            <field name='keyOutput' type='MFString' accessType='outputOnly'/>
            <field name='stringOutput' type='MFString' accessType='outputOnly'/>
<![CDATA[
ecmascript:

function keyInput (inputValue)
{
    Browser.print ('keyInput=' + inputValue + '\n'); // console output
    keyOutput = new MFString (inputValue); // type conversion
}
function finalTextInput (inputValue)
{
    Browser.print ('finalText=' + inputValue + '\n'); // console output
    stringOutput = new MFString (inputValue); // type conversion
}
function enteredTextInput (inputValue)
{
    Browser.print ('enteredText=' + inputValue + '\n'); // console output
}
]]>
            </Script>
            <ROUTE fromNode='DefaultKeySensor' fromField='keyPress' toNode='KeyboardProcessor' toField='keyInput'/>
            <ROUTE fromNode='DefaultStringSensor' fromField='finalText' toNode='KeyboardProcessor' toField='finalTextInput'/>
            <ROUTE fromNode='DefaultStringSensor' fromField='enteredText' toNode='KeyboardProcessor' toField='enteredTextInput'/>
            <ROUTE fromNode='KeyboardProcessor' fromField='keyOutput' toNode='KeyText' toField='string'/>
            <ROUTE fromNode='KeyboardProcessor' fromField='stringOutput' toNode='StringText' toField='string'/>
```

123:1 INS

Note that a Script node is needed to convert the SFString outputs of the KeySensor and TouchSensor into MFString inputs for the appropriate Text node *string* field.

This is one of the few remaining cases in X3D where a Script node is needed for data type conversion between a sensor output node and another X3D target node.

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter08-UserInteractivity/UserInteractivitySensorNodes.x3d

back to Table of Contents

Chapter Summary

60

Summary: User Interactivity

User interactivity is initiated via sensor nodes, which capture user inputs and are hooked up to provide appropriate responses

- TouchSensor senses pointing device (mouse, etc.)
- PlaneSensor is a drag sensor that converts x-y pointer motion to move objects in a plane
- CylinderSensor and SphereSensor are drag sensors that convert x-y pointer motion to rotate objects
- KeySensor and StringSensor capture keyboard input

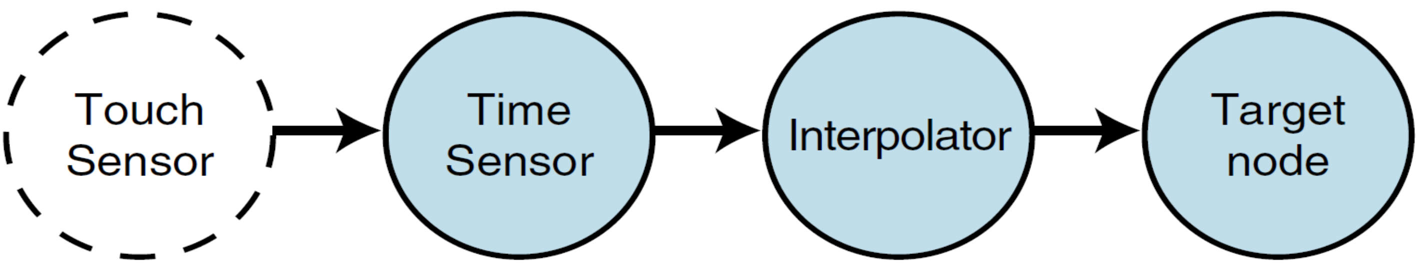
Interactivity sensors initiate animation chains

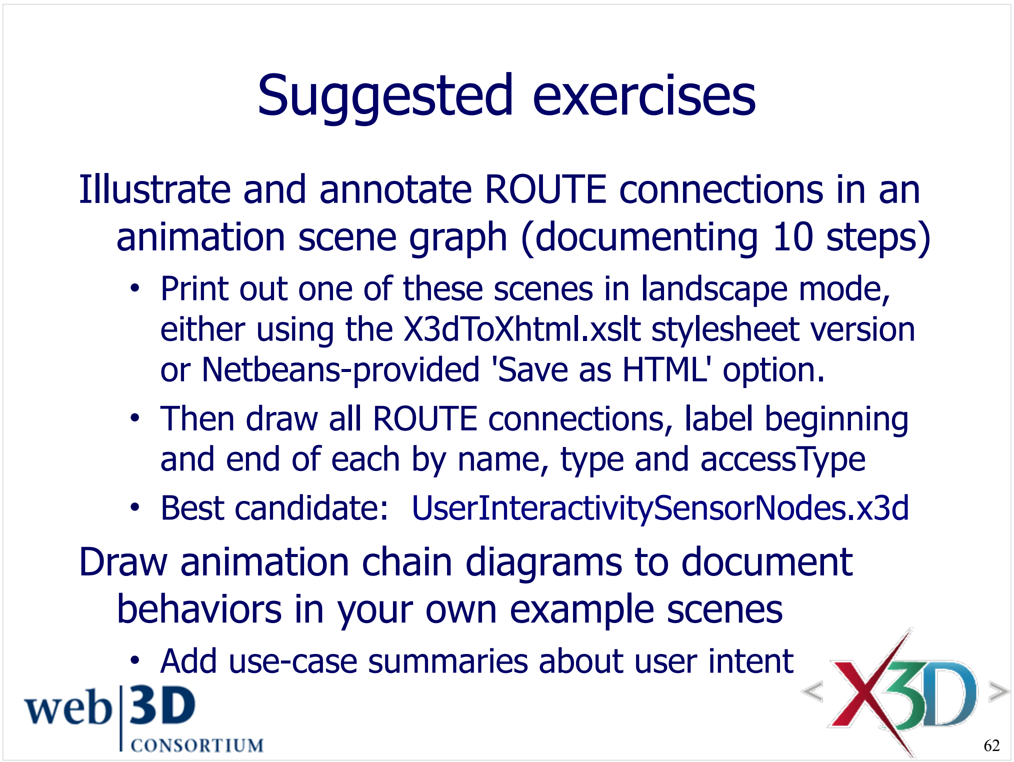
web3D CONSORTIUM

61

Dragging is the movement of a selected object using the pointing device, a capability provided by the drag sensors.

Animation chains are covered in Chapter 7, Event Animation and Interpolation.



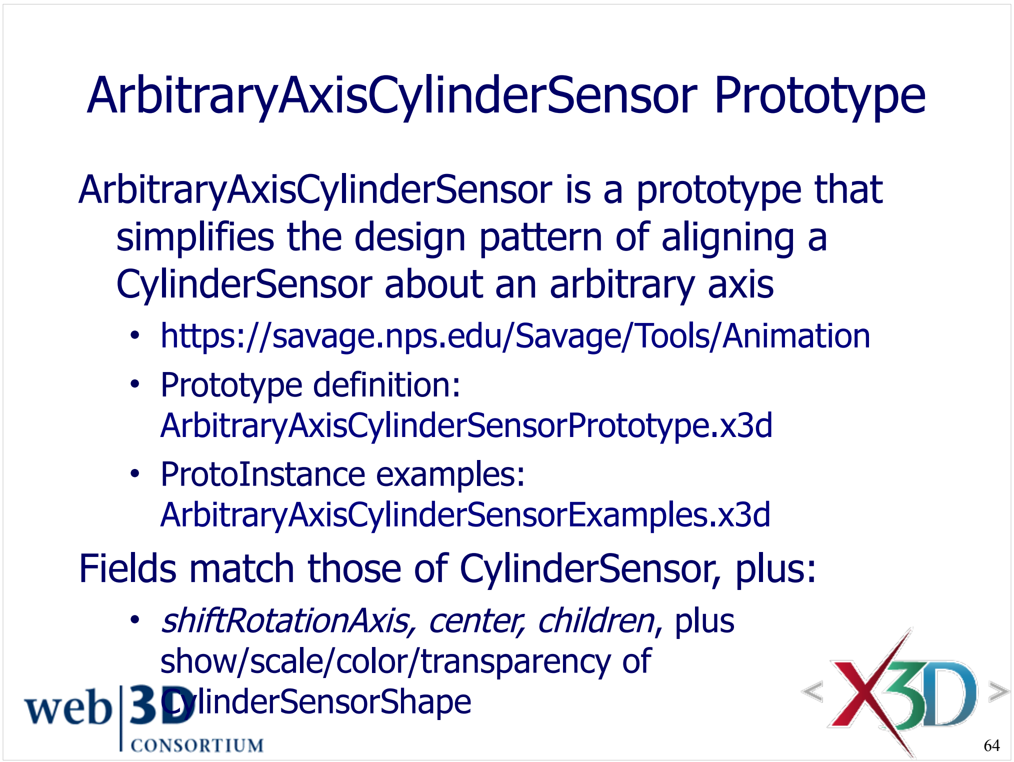


Someday we hope to automate the production of such diagrams.

X3dToXhtml.xslt is available via X3D-Edit menu *X3D, Conversions*

back to Table of Contents

Additional Resources



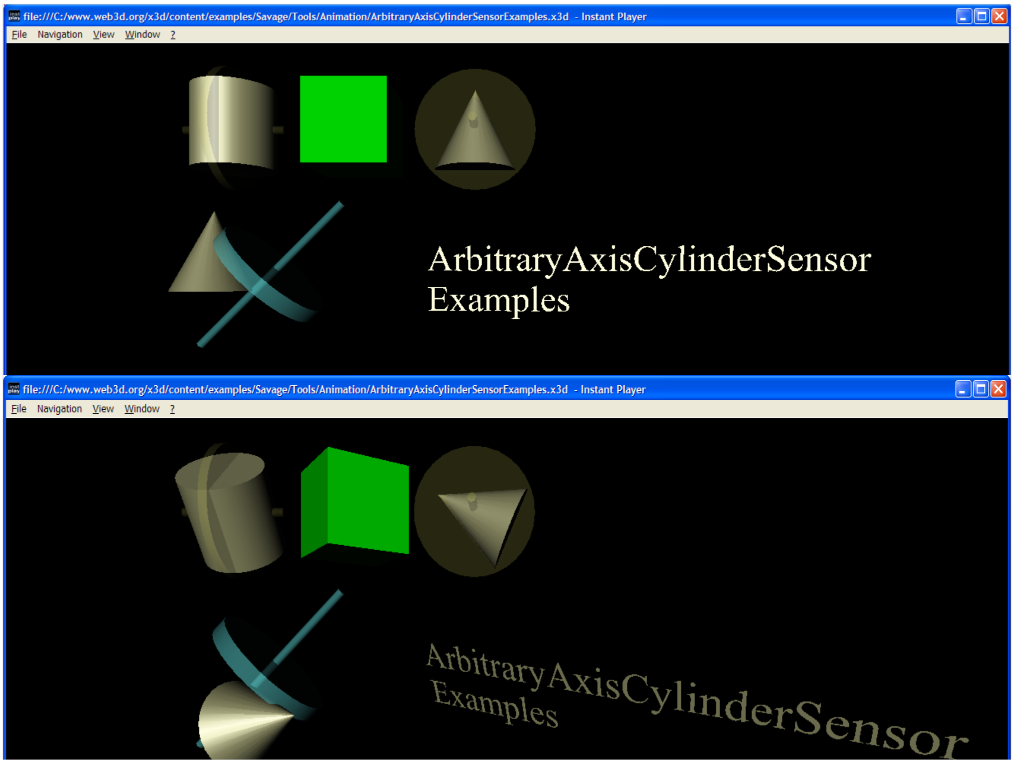
Prototypes are an extensibility mechanism to define new X3D nodes using existing X3D nodes. They are covered in Chapter 14.

Warning: ArbitraryAxisCylinderSensor operates on its children, NOT on its peers. This variation is necessary in order to accomplish the desired Transform rotation to a new orientation axis. Example use:

https://savage.nps.edu/Savage/Tools/Animation/ArbitraryAxisCylinderSensorExamples.x3d

```
<ExternProtoDeclare name="ArbitraryAxisCylinderSensor">
        <!-- copy field definitions here -->
</ExternProtoDeclare>

<ProtoInstance name='ArbitraryAxisCylinderSensor' containerField='children'>
        <!-- rotate rotate CylinderSensor yAxis to xAxis -->
        <fieldValue name='shiftRotationAxis' value='0 0 1 -1.5707963'/>
        <fieldValue name='children'>
                <Shape>
                        <Cylinder/>
                        <Appearance>
                                <Material diffuseColor='1 0 0'/>
                        </Appearance>
                </Shape>
        </fieldValue>
</ProtoInstance>
```



These screen snapshots show the original unmanipulated scene above, and multiple user-rotated objects with different axis angles in the scene below.

https://savage.nps.edu/Savage/Tools/Animation/ArbitraryAxisCylinderSensorExamples.x3d

DoubleClickTouchSensor

DoubleClickTouchSensor is a prototype alternative to TouchSensor that detects when a user has rapidly selected an object twice

- https://savage.nps.edu/Savage/Tools/Animation
- Prototype definition: DoubleClickTouchSensorPrototype.x3d
- ProtoInstance examples: DoubleClickTouchSensorExample.x3d

Fields match those of TouchSensor, plus:

- *maxDelayInterval* allowed for distinguishing between single and double click, in seconds

web3D CONSORTIUM

66

Prototypes are an extensibility mechanism to define new X3D nodes using existing X3D nodes. They are covered in Chapter 14.

Example use:

https://savage.nps.edu/Savage/Tools/Animation/DoubleClickTouchSensorExample.x3d

```
<ExternProtoDeclare name=”DoubleClickTouchSensor”>
            <!-- copy field definitions here -->
</ExternProtoDeclare>

<ProtoInstance name='DoubleClickTouchSensor' DEF='TouchSensorActive'>
   <fieldValue name='description'
             value='double click to initiate time delay and color change'/>
   <fieldValue name='maxDelayInterval' value='0.5/>
</ProtoInstance>
```

TimeDelaySensor Prototype

TimeDelaySensor is an alternative to TimeSensor that includes a time delay before firing

- https://savage.nps.edu/Savage/Tools/Animation
- Prototype definition: TimeDelaySensorPrototype.x3d
- ProtoInstance examples: TimeDelaySensorExample.x3d

Fields match those of TimeSensor, plus:

- *delayInterval*, *delayCompleteTime*

web3D CONSORTIUM

67

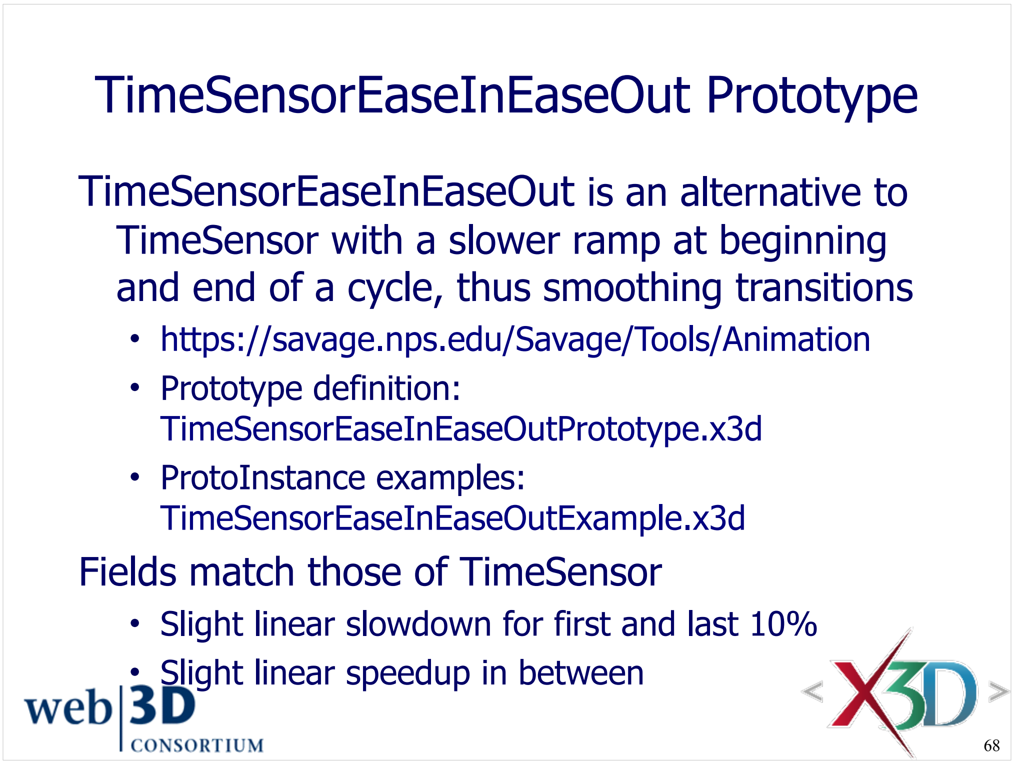
Prototypes are an extensibility mechanism to define new X3D nodes using existing X3D nodes. They are covered in Chapter 14.

Example use:

https://savage.nps.edu/Savage/Tools/Animation/TimeDelaySensorExample.x3d

```
<ExternProtoDeclare name='TimeDelaySensor'
        url='"TimeDelaySensorPrototype.x3d#TimeDelaySensor"
"https://savage.nps.edu/Savage/Tools/Animation/TimeDelaySensorPrototype.x3d #TimeDelaySensor"
"TimeDelaySensorPrototype.wrl#TimeDelaySensor"
"https://savage.nps.edu/Savage/Tools/Animation/TimeDelaySensorPrototype.wrl#TimeDelaySensor"'>
    <field accessType='inputOutput' name='startTime' type='SFTime'/>
    <field accessType='inputOutput' name='enabled' type='SFBool'/>
    <field accessType='inputOutput' name='delayInterval' type='SFTime'/>
    <field accessType='outputOnly' name='delayCompleteTime' type='SFTime'/>
    <field accessType='initializeOnly' name='traceEnabled' type='SFBool'/>
</ExternProtoDeclare>

<ProtoInstance DEF='DelayTimer' name='TimeDelaySensor'>
  <fieldValue name='delayInterval' value='3'/>
  <fieldValue name='traceEnabled' value='true'/>
</ProtoInstance>
```

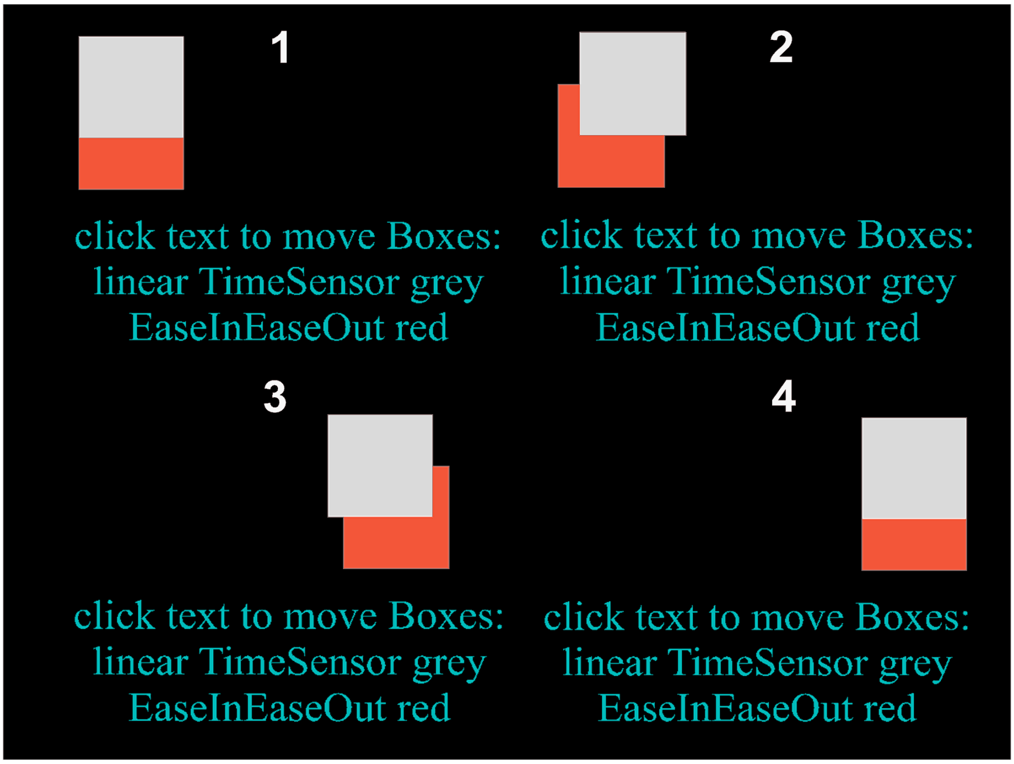


Prototypes are an extensibility mechanism to define new X3D nodes using existing X3D nodes. They are covered in Chapter 14.

Example use:
https://savage.nps.edu/Savage/Tools/Animation/TimeSensorEaseInEaseOutExample.x3d

```
<ExternProtoDeclare name='TimeSensorEaseInEaseOut'>
        <!-- need to copy url and field definitions here -->
</ExternProtoDeclare>

<ProtoInstance name='TimeSensorEaseInEaseOut' DEF='EasyClock'>
    <fieldValue name='cycleInterval' value='3'/>
</ProtoInstance>
```



Snapshots showing progression of a TimeSensorEaseInEaseOut animation.

Each box starts and stops at the same locations and also at the same times. The white TimeSensor box travels at a constant speed throughout.

The TimeSensorEaseInEaseOut orange box starts more slowly at the start, speeds up to pass the white box, then slows to finish identically. This can be a more graceful way to perform some animations.

https://savage.nps.edu/Savage/Tools/Animation/TimeSensorEaseInEaseOutExample.x3d

back to Table of Contents

References

70

References 1

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

- Chapter 8, User Interactivity
- http://x3dGraphics.com
- http://x3dgraphics.com/examples/X3dForWebAuthors

X3D Resources

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

web 3D CONSORTIUM

71

References 2

X3D-Edit Authoring Tool

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

X3D Scene Authoring Hints

- http://x3dgraphics.com/examples/X3dSceneAuthoringHints.html

X3D Graphics Specification

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

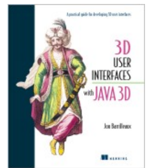
web3D CONSORTIUM

72

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 9 - Sensing Viewer



3D User Interfaces with Java3D by Jon Barilleaux, Manning Publications, 2001.

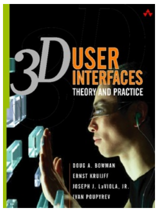
- http://www.manning.com/barrilleaux
- http://java.sun.com/developer/Books/Java3D

web3D CONSORTIUM

73

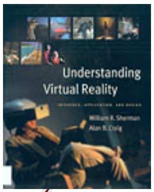
References 4

3D User Interfaces: Theory and Practice by Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola Jr. and Ivan Poupyrev, Addison Wesley, 2005.



- http://www.3dui.org
- http://people.cs.vt.edu/~bowman/3dui.org/3D UI Book.html

Understanding Virtual Reality: Interface, Application and Design by Bill Sherman and Alan Craig, Morgan Kaufmann, 2003.



- http://www.immersence.com/publications/2003/2003-WSherman.html

web3D CONSORTIUM

X3D

Conferences 1

ACM SIGGRAPH

- Special Interest Group on Graphics is the leading professional society for computer graphics and interactive techniques
- http://www.siggraph.org

ACM SIGCHI

- Special Interest Group on Computer-Human Interaction, brings together people working on the design, evaluation, implementation, and study of interactive computing systems for human use
- http://www.sigchi.org

web3D CONSORTIUM

X3D

75

Conferences 2

IEEE Symposium on 3D User Interfaces (3DUI)

- http://conferences.computer.org/3dui

IEEE Symposium on Virtual Reality (VR)

- http://conferences.computer.org/vr

Web3D Symposium

- In cooperation with Web3D Consortium, ACM SIGGRAPH and Eurographics
- http://www.web3d2009.org

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

web3D CONSORTIUM

77

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:

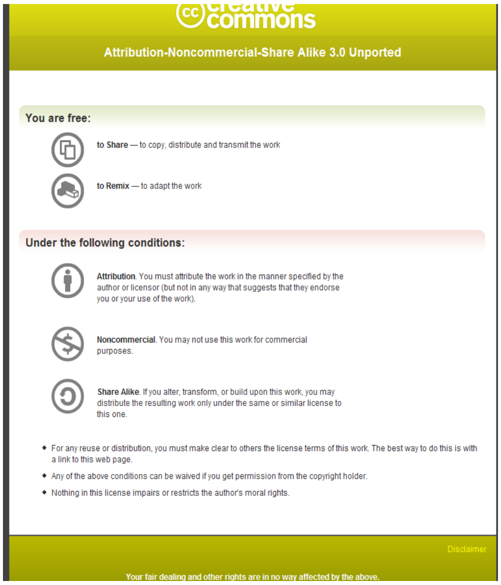
- http://cgems.inesc.pt/authors/ListModules.aspx

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

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