Chapter 4

Viewing and Navigation

But the eyes, though they are no sailors, will never be satisfied with any model, however fashionable, which does not answer all the requisitions of art.

Henry David Thoreau, 1849
Contents

Chapter Overview and Concepts

X3D Nodes and Examples

Additional Resources

Chapter Summary and Suggested Exercises

References
Chapter Overview
Overview: Viewing and Navigation

Users explore X3D worlds by choosing predefined viewpoints and navigating through 3D space.

- Bindable nodes, so only one is active at a time
- Viewpoint lets authors identify key camera locations
- NavigationInfo provides options for moving around

Related nodes improve navigability, interaction

- Anchor makes geometric shapes linkable
- Billboard keeps child geometry facing the user
- Collision can allow or prevent a user's view from passing through geometry
Concepts
Viewing and navigation

It is helpful to think of X3D scenes as fixed at different locations in 3D space

• Viewpoints are like cameras, prepositioned in locations (and directions) of interest
• Users can move their current camera viewpoint further and change direction they are looking at
• This process is called navigation

Making navigation easy for users is important

• Authors provide viewpoints of interest with scenes
• Browsers enable camera rotation, pan, zoom, etc.
Goals of viewing and navigation

• Viewing a scene from different vantage points that reveal aspects of interest, document key locations, or help to tell a story

• Navigating changes in the user’s view of a scene effectively, by moving from place to place in an intuitive manner

• Making geometric objects selectable so that users can transport to another viewpoint, launch into another scene, or receive other web content

• Taking advantage of viewpoint location for special interactive techniques, such as user-facing billboard rotations and terrain following
Bindable nodes

Bindable nodes have a special property: only one can be active at a time.

-Bindable nodes are Viewpoint, NavigationInfo, Background, TextureBackground, Fog
-Each implements X3DBindableNode type interface for consistency
-First nodes found in scene become active by default

Implemented using a stack

-Similar to spring-loaded tray of plates in cafeteria
-One (and only one) is active, on top
-One can be pulled off top, sent off to the side
-One can be pulled to top, pushing down others
Binding example

Basic user operation is pretty simple:

• just select the desired Viewpoint

Complex example follows, stepping through binding stack operations

• Advanced details
• BindingOperations.x3d
• Animated with scripting
• Console results found in BindingOperations.console.txt

New users please skip ahead to Nodes and Examples
Binding node operations diagram

Viewpoints are activated (bound) upon selection, ordering is governed by stack operations.
Binding node operations

- **Time t0.** The initial loading of the scene has first `<Viewpoint DEF='View1'/>` active and bound to the top of the binding stack. Other viewpoints are off the binding stack.

- If no viewpoints are provided in the scene, then the default `<Viewpoint position='0 0 10'/>` defined in the X3D Specification is used.

- **Time t1.** When the user selects View2 from the viewpoint list, it receives a `set_bind="true"` event and goes to the top of the binding stack. View2 also issues an `isBound="true"` event, and View1 issues an `isBound="false"` event as it moves down the stack.
Binding node operations 2

- **Time t2.** Similar to the previous transitions in step t1, View3 receives a set_bind="true" event and responds with an isBound="true" event, while View2 issues an isBound="false" event and pushes View1 further down the stack.

- **Time t3.** View3 receives a set_bind="false" event, triggering a corresponding isBound="false" event and dropping off the stack completely. Because View2 is the next node on the binding stack, it pops to the top to become the active Viewpoint node. View2 also issues an isBound="true" event.
Binding node operations

- **Time t4.** The user now selects View1 from the browser’s viewpoint list, so View1 receives a `set_bind="true"` event and sends a corresponding `isBound="true"` event. View2 is no longer bound, and is pushed down the binding stack.

- **Time t5.** View2 receives a `set_bind="false"` event while on the binding stack but unbound, and as a result, it is taken completely off the binding stack.

- **Time t6.** View1 is now removed off the binding stack via a `set_bind="false"` event, leaving no other defined Viewpoint nodes on the stack.
Binding node operations

- **Time t7.** With no Viewpoint nodes remaining on the stack to bind, default viewpoint values are used: `<Viewpoint position='0 0 10'/>`. The user then selects the previously unbound View4 from the viewpoint list.

- **Time t8.** View4 remains as the bound viewpoint with no further viewpoints remaining on the stack.

Same process for all X3D bindable node stacks:

- Viewpoint/OrthoViewpoint/GeoViewpoint, NavigationInfo/GeoViewpoint, Fog, Background/TextureBackground
X3D Nodes and Examples
Viewpoint node

It is helpful to think of X3D scenes as being fixed solidly in 3D space, positioned and oriented exactly where placed by the scene author.

Viewing a scene is thus a matter of navigating the current user point of view through space.

Viewpoint nodes let X3D scene authors predefine locations and orientations of particular interest:

- Sometimes viewpoints are animated and moving.
- Freedom of viewpoint is exciting and engaging, also a major advantage over fixed-viewpoint video.
Viewpoint list

Viewpoint list is optional browser-provided feature that lists currently available viewpoints

- Provides description information for viewpoints
- Simplifies user selection of viewpoints
- Thus supports navigation within a scene

Viewpoints are listed in the order that they appear in the “extended scene graph”

- First includes order of definition in primary scene
- Then includes viewpoints provided by Inline and prototype instances, inserted in order within the list
- Authors need to order Viewpoints carefully so that user navigation, understanding is best supported
Viewpoint *description*

Each Viewpoint is given a *description* string to help users decide which view to select

- Clear, understandable descriptions can guide users
- Use an object's name first when many viewpoints follow, so they are more easily identified in a list
- Use whitespace instead of underscores for better readability

Viewpoints are primary user tool for navigation

- Browsers provide Viewpoint List to show and select descriptions
- So authors should always include *description*!
Viewpoint *position, orientation*

A Viewpoint node defines a specific *position* and *orientation* for looking at a 3D scene

• Similar to a “virtual camera” vantage point

Default Viewpoint *position* is (0 0 10)

• out 10 m on +Z axis, looking back towards origin

Any changes to Viewpoint *orientation* are made relative to that default direction (along -Z axis)

• Different initial direction than other orientations
• Visualize the situation and then use right-hand rule to figure out the correct *orientation* value
Viewpoint calculator

Viewpoint Calculator provided as an author-assist tool to compute Viewpoint orientation to a given point

Inputs:
- position
- goal look-at point
- twist angle about viewing axis

Output:
- orientation
Viewpoint *centerOfRotation*, *fieldOfView*

*centerOfRotation* is a local position
- User's current view rotates about this point if the bound NavigationInfo node is in EXAMINE mode
- Can be changed by a user's LOOKAT operation picking some other geometry as new center

*fieldOfView* is preferred minimum angular width
- Shorter side of horizontal width or vertical height
- Default is 45 degrees = \( \pi/4 \) radians = 0.785
- Larger side determined by browser aspect ratio
- Author can set width, height if within HTML page
ViewFrustum prototype

ViewFrustum is a helpful visualization prototype

- Prototypes simplify creation of new X3D nodes
- Shows near and far clipping planes that truncate the viewable area
- Depends on Viewpoint and NavigationInfo parameters

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

\[
\text{nearHalfWidth} = \tan\left(\frac{\text{fieldOfView}}{2}\right) \times \text{avatarSize}[0]; \\
\text{farHalfWidth} = \tan\left(\frac{\text{fieldOfView}}{2}\right) \times \text{visibilityLimit};
\]
**Viewpoint jump**

*jump* can be a tricky field (but is not often used)

- *jump*='true' when a Viewpoint is selected means that the current view position and orientation is modified according to NavigationInfo *transitionType*
- *jump*='true' is usual default
- *jump*='false' is an advanced technique
  - User's view doesn't appear to change when new Viewpoint is selected
  - New Viewpoint is bound, but given offsets to match prior user position and orientation (hence no jumping)
  - Example use: changing bound viewpoint when moving from one floor into an elevator, then to another floor
Viewpoint hints and warnings

Use parent Transform node(s) for complex Viewpoint orientation and position values
  • One axis of rotation at a time can work more clearly

Keyboard shortcuts are helpful
  • PageUp PageDown Home End to select Viewpoint
  • Arrow keys to examine (rotate), pan, zoom, etc. depending on current NavigationInfo mode
  • Browser may allow Viewpoint reset after navigating

Distinguish between defined Viewpoint and current navigated user-view location, direction

web3D consortium
Illustrate Viewpoint binding operations as described in Chapter 4 concepts. Display the browser console to see an event.

Viewpoint node X3D-Edit

View #1
View #2
View #3
View #4

Click here to animate
X3D for Web Authors, Kelp Forest Exhibit: Kelp Forest Main

This is the main entry to the Kelp Forest that loads all of the other components as Inline scenes.
Welcome to the NPS simulation of the Monterey Bay Aquarium Kelp Forest

Find sharks! See new viewpoints! Press PageDown, wait and watch.
Welcome to the NPS simulation of the Monterey Bay Aquarium Kelp Forest

Find sharks! See new viewpoints! Press PageDown, wait and watch.
Viewpoint provides a specific location and direction where the user may view the scene. Background, Fog, NavigationInfo, TextureBackground and Viewpoint are bindable nodes.

**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 initializeOnly, type SFSFString CDATA #IMPLIED]
Text description or navigation hint to be displayed for this Viewpoint.

**Hint:** use spaces, make descriptions clear and readable.

**Warning:** without description, Viewpoint is unlikely to appear on browser Viewpoints menu. Hint: many XML tools substitute XML character references automatically if needed (like &amp;#38; for & and &amp;#34; for ”).

**Interchange profile hint:** this field may be ignored.

**position**

[position: accessType inputOutput, type SFVec3f CDATA "0 0 10"]
Position (x, y, z in meters) relative to local coordinate system.

**orientation**

[orientation: accessType inputOutput, type SFRotation CDATA "0 0 1 0"]
Rotation (axis, angle in radians) of Viewpoint, relative to default -Z axis direction in local coordinate system.

**Hint:** this is orientation _change_ from default direction (0 0 -1).

**Hint:** complex rotations can be accomplished axis-by-axis using parent Transforms.

**fieldOfView**

[fieldOfView: accessType inputOutput, type SFFloat CDATA "0.785398" (0..pi)]
Preferred minimum viewing angle from this viewpoint in radians. Small field of view roughly corresponds to a telephoto lens, large field of view roughly corresponds to a wide-angle lens.

**Hint:** modifying Viewpoint distance to object may better be for zooming.

**Warning:** fieldOfView may not be correct for different window sizes and aspect ratios.

**Interchange profile hint:** this field may be ignored.

**jump**

[jump: accessType inputOutput, type SFBool (true|false) "true"]
Transition instantly by jumping, or smoothly animate to this Viewpoint.

**Hint:** set jump=true for smooth camera motion when going to this viewpoint.

**centerOfRotation**

[centerOfRotation: accessType inputOutput, type SFVec3f CDATA "0 0 0"]
CenterOfRotation point relates to NavigationInfo EXAMINE mode.

**set_bind**

[set_bind: accessType inputOnly, type SFBool (true|false) #FIXED ""]
Sending event set_bind=true makes this node active. Sending event set_bind=false makes this node inactive. Thus setting set_bind to true/false will pop/push (enable/disable) this Viewpoint.

**bindTime**

[bindTime: accessType outputOnly, type SFTime CDATA #FIXED ""]
Event sent when node becomes active/inactive.

**isBound**

[isBound: accessType outputOnly, type SFBool (true|false) #FIXED ""]
Event true sent when node becomes active, event false sent when unbound by another node.

**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.
OrthoViewpoint

OrthoViewpoint provides an orthographic perspective-free view of a scene from a specific location and direction

- **fieldOfView** minX, maxX, minY, maxY values (default -1 -1, 1 1) define minimum and maximum extents of view, provided in units of local coordinate system

- For a rectangular display:
  
  width/height = (maxX-minX)/(maxY-minY)

- `<component name='Navigation' level='3'/>`

- Caution: often disorienting, special uses only
Navigation model 1

Users can select predefined Viewpoints
  • Defines both position and direction of view

Users can further navigate around scene
  • Using pointing device or hot keys
  • Chosen viewpoint remains bound

<table>
<thead>
<tr>
<th>Key</th>
<th>Emulated Action</th>
<th>WALK mode</th>
<th>FLY mode</th>
<th>EXAMINE mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up arrow</td>
<td>Pointer up</td>
<td>forward</td>
<td>forward</td>
<td>orbit up</td>
</tr>
<tr>
<td>Down arrow</td>
<td>Pointer down</td>
<td>backward</td>
<td>backward</td>
<td>orbit down</td>
</tr>
<tr>
<td>Left arrow</td>
<td>Pointer left</td>
<td>left</td>
<td>left</td>
<td>orbit left</td>
</tr>
<tr>
<td>Right arrow</td>
<td>Pointer right</td>
<td>right</td>
<td>right</td>
<td>orbit right</td>
</tr>
</tbody>
</table>

These are the default navigation key responses
Navigation model 2

User's current view can itself be animated

• ROUTE new position/direction event values to the Viewpoint itself, or to parent Transform nodes
• User navigation offsets to that view remain in effect
• Thus “over the shoulder” viewpoints can follow a moving object around, while still allowing user to look around while in that moving viewpoint

Lefty and Lucy shark in the Kelp Forest Main scene use this technique as virtual tour guides
NavigationInfo node

NavigationInfo indicates how a browser might best support user navigation in the scene

Multiple NavigationInfo nodes may exist in scene
  • Or in multiple Inline scenes loaded together

NavigationInfo is an X3DBindableNode
  • So only one NavigationInfo can be active at a time
  • Follows the same binding rules as Viewpoint, but note that they are easily selectable by end users
  • Can be linked to a given Viewpoint by ROUTE that connects isBound of one node to set_bind of other
NavigationInfo type

Primary field is *type* which indicates which of the various modes of navigation are relevant:

- "EXAMINE" best for rotating solitary objects
- "FLY" allows zooming in, out and around
- "WALK" also allows exploration, but on the ground
- "LOOKAT" use pointer to select geometry of interest
- "ANY" lets user select any mode
- "NONE" gives user zero control of navigation

MFString array default *type*=' "EXAMINE" "ANY" ' • which gives users plenty of flexibility
NavigationInfo type details

• "EXAMINE" Used to view individual objects. Scene navigation consists of rotating the user viewpoint about the center of the observed object. The `centerOfRotation` field of the currently bound Viewpoint node values determines which local point centers the view rotation.

• "WALK" Used when exploring a virtual world on the ground. The user’s eye level stays above the ground geometry and collision detection prevents the user from falling if underlying geometry is present.
NavigationInfo *type* details 2

- **"FLY"** Similar to "WALK", but terrain following and collision detection is ignored. This type of navigation has the fewest constraints. Shifts the current view and related `centerOfRotation` values to track or zoom toward objects of interest to user.
- **"ANY"** Browser is allowed to provide whichever navigation type seems appropriate for the task at hand, modifying the user interface if necessary.
- **"NONE"** All navigation disabled and hidden. Navigation remains possible via animation of viewpoint fields or by binding other viewpoints (using viewpoint-list selection or Anchor node).
NavigationInfo speed, headlight

**speed** determines how fast navigation occurs
- Default value 1 meter/second is usually pretty slow
- Might need to vary widely from ground to space
- Might need multiple NavigationInfo nodes matching different viewpoints (high **speed** for flying, low **speed** for walking around or examining objects)

**headlight** is whether a light is shining ahead from user's point of view
- Otherwise one or more Light nodes is needed (covered in Chapter 11), or else world goes black
NavigationInfo transitions

transitionType determines type of path followed when transitioning between viewpoints

- "ANIMATE" browser chooses smoothing algorithm
- "LINEAR" interpolation of position, orientation
- "TELEPORT" immediate repositioning to destination

transitionTime

- initial array value used for linear, otherwise multiple values can be used by browser-specific "ANIMATE"

transitionComplete lets author know when done

- SFBool boolean event sent when move is finished
avatarSize SFVec3f array

avatarSize[0] = 0.25m
Allowed collision distance

<NavigationInfo avatarSize='0.25 1.6 0.75'/>

avatarSize[1] = 1.6m
Viewpoint height above terrain

avatarSize[2] = 0.75m
Maximum step-over height

Y=0
Local ground level
NavigationInfo \textit{visibilityLimit}\\
\textit{visibilityLimit} defines the maximum range that may be rendered by the browser\\
  \begin{itemize}
  \item Measured from the user's point of view
  \item Geometry beyond that distance are not drawn
  \item \textit{visibilityLimit}='0.0' means no limits are imposed
  \end{itemize}

Quality thumbrule: meet following relationship\\
  \begin{itemize}
  \item \textit{avatarSize-collisionDistance} / \textit{visibilityLimit} < 10,000
  \item Avoids floating-point roundoff error on graphics card and almost-coplanar polygon tearing/aliasing
  \item Exactly coplanar polygons still suffer from aliasing
<?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='NavigationInfo.x3d' name='title'/>
    <meta content='NavigationInfo override example for Kelp Forest world.' name='description'/>
    <meta content='Leonard Daly and Don Brutzman' name='creator'/>
    <meta content='14 November 2005' name='created'/>
    <meta content='6 January 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) 2005, Daly Realism & Don Brutzman' name='rights'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter04-ViewingNavigation/NavigationInfo.x3d'
name='identifier'/>
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='..//license.html' name='license'/>
  </head>
  <Scene>
    <!-- Because this NavigationInfo node is in the parent scene, it is bound at load time, and thus governs the active navigation modes -->
    <NavigationInfo type="EXAMINE" ANY"/>
    <Inline url='"../../../KelpForestExhibit/KelpForestMain.x3d"
    "../../../KelpForestExhibit/KelpForestMain.wrl"
    "http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit"
    "http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit"
    "http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit"
    />
  </Scene>
</X3D>
**NavigationInfo**

NavigationInfo describes the viewing model and physical characteristics of the viewer's avatar. The type ""NAVIGATION"" places additional requirements on the viewing model and geometry. Use ""NAVIGATION"" for applications that require precise camera-to-object collision detection. Background, Fog, NavigationInfo, TextureBackground and Viewpoint are bindable nodes.

### DEF

**DEF ID #IMPLIED**

DEF defines a unique ID name for this node, referencable by other nodes. Hist: 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. Hist: 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!

### type

**[type: accessType inputOutput, type MFString CDATA "EXAMINE" "ANY"]**

Enter one or more quoted Strings: "EXAMINE" "WALK" "FLY" "LOOKAT" "ANY" "NONE". Hist: for inspection of simple objects, usability often improves with type="EXAMINE" "ANY". Hist: types WALK and FLY force strict camera-to-object collision detection. Hist: see Collision node for further details on camera-to-object collision detection. Hist: Strings can have multiple values, so separate each string by quote marks ['http://www.url1.org, http://www.url2.org' etc.]

Interchange profile hint: this field may be ignored.

### speed

**[speed: accessType inputOutput, type SFFloat CDATA "1.0"]**

0.0 to infinity Default rate at which viewer travels through scene, meters/second. Warning: default 1 m/s usually seems slow for ordinary navigation.

Interchange profile hint: this field may be ignored.

### headlight

**[headlight: accessType inputOutput, type SFBool (true/false) "true"]**

Enable/disable directional light that always points in the direction the user is looking.

### avatarSize

**[avatarSize: accessType inputOutput, type MFFloat CDATA "0.25 1.6 0.75"]**

avatarSize triplet values are: (a) collision distance between user and geometry (near culling plane of the view frustrum) (b) viewer height above terrain (c) tallest height viewer can WALK over. Hist: keep (visibilityLimit / avatarSizeCollisionDistance) < 10,000 to avoid aliasing artifacts (i.e. polygon "tearing").

Interchange profile hint: this field may be ignored.

### visibilityLimit

**[visibilityLimit: accessType inputOutput, type SFFloat CDATA "0.0"]**

Geometry beyond the visibilityLimit may not be rendered (far culling plane of the view frustrum). visibilityLimit=0.0 indicates an infinite visibility limit. Hist: keep visibilityLimit >= zero. Hist: keep (visibilityLimit / avatarSizeCollisionDistance) < 10,000 to avoid aliasing artifacts (i.e. polygon "tearing").

Interchange profile hint: this field may be ignored.

### transitionType

**[transitionType: accessType inputOutput, type MFString CDATA "ANIMATE"]**

Enter one or more quoted Strings: "ANIMATE" "LINEAR" "TELEPORT".

Interchange profile hint: this field may be ignored.

### transitionTime

**[transitionTime: accessType inputOutput, type MFFloat CDATA "1.0"]**

Duration of viewpoint transition. Hist: If transitionType is "ANIMATE", transitionTime provides browser-dependent animation parameters.

Interchange profile hint: this field may be ignored.

### transitionComplete

**[transitionComplete: accessType outputOnly, type MFFloat CDATA #FIXED ""]**

Event signaling viewpoint transition complete.

Interchange profile hint: this field may be ignored.

### set_bind

**[set_bind: accessType inputOnly, type SFBool (true/false) #FIXED ""]**

Setting set_bind true makes this node active setting set_bind false makes this node inactive. Thus setting set_bind true/false will pop/push (enable/disable) this node.

### bindTime

**[bindTime: accessType outputOnly, type SFFloat CDATA #FIXED ""]**

Event sent when node becomes active/inactive.

### inBound

**[inBound: accessType outputOnly, type SFBool (true/false) #FIXED ""]**

Event true sent when node becomes active, event false sent when unbound by another node.

### 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.
Aliasing (tearing) of polygons
Anchor node

Anchor is another grouping node that can contain other nodes.

Geometry rendered by contained nodes is active and can be selected by user:

- User clicking on Anchor geometry launches url link (.html .x3d mailto: etc.)
  - thus similar to HTML anchor tag `<a href="url">link</a>`
- Alternatively can select a viewpoint in the scene (similar to HTML bookmark)

Selected link can replace current X3D scene, or else launch into another browser window.
Anchor *description*

The *description* field provides the user with a single-string summary of what is selected when the Anchor geometry is selected, e.g.

- *description*='click door, open portal to new world'
- *description*='jump to next viewpoint...'

X3D browsers usually pop up the text description when the pointing device is over the selection geometry.
url Uniform Resource Locator

The *url* field provides either

- Address to new X3D scene, HTML page, or another Web resource, or else
- Viewpoint bookmark within the scene

**MFString array provides alternate url addresses**

- url addresses can be either local or online
- Point to alternate versions of same resource
- X3D browser goes sequentially through ordered list, one at a time, until one retrieval succeeds
Anchor *parameter*

*parameter* provides additional information to browser regarding redirection of loaded result

- *parameter*='target=_blank' sends to new frame
- *parameter*='target=frame4' sends to named frame
- May be ignored if browser is solely X3D capable, rather than (for instance) a Web-browser plugin

Once again, designed to match functionality found in HTML anchor tag
Anchor hints and warnings

Strictly match capitalization of directories and file names

- Unix and http are case sensitive and fail otherwise
- Windows is forgiving but actually this hides errors

XML escape characters

- & (ampersand) \&amp;
- ' (apostrophe) \&apos;
- " (double quote) \&quot;
Click orange text to launch Monterey Bay Aquarium kelp forest HTML page

Click yellow text to move to a diagonal Viewpoint

Click green text to load X3D Coordinate Axes scene
Anchor example scene, editor

Welcome to the NPS simulation of the Monterey Bay Aquarium Kelp Forest
Click orange text for Monterey Bay Aquarium kelp forest web site
Anchor.x3d - Editor

Anchor.x3d  scene,  X3D-Edit
### Anchor

Anchor is a Grouping node that can contain most nodes. Clicking Anchored geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is same window.

**Hint:** insert a Shape node before adding geometry or Appearance.

**DEF**

<table>
<thead>
<tr>
<th>DEF ID</th>
<th>#IMPLIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF</td>
<td>defines a unique ID name for this node, referenceable by other nodes.</td>
</tr>
<tr>
<td>Hint</td>
<td>descriptive DEF names improve clarity and help document a model.</td>
</tr>
</tbody>
</table>

**USE**

<table>
<thead>
<tr>
<th>USE IDREF</th>
<th>#IMPLIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE</td>
<td>means reuse an already DEF-ed node ID, ignoring <em>all</em> other attributes and children.</td>
</tr>
<tr>
<td>Hint</td>
<td>USEing other geometry (instead of duplicating nodes) can improve performance.</td>
</tr>
</tbody>
</table>

**Warning:** do NOT include DEF (or any other attribute values) when using a USE attribute!

**description**

<table>
<thead>
<tr>
<th>description: accessType inputOutput, type SFSString CDATA</th>
<th>#IMPLIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text description to be displayed for action of this node.</td>
<td>Hint: many XML tools substitute XML character references automatically if needed (like &quot;&amp;&quot;, for &quot;&amp;&quot; or &quot;&quot;&quot;, for &quot;)</td>
</tr>
<tr>
<td>Interchange profile hint: this field may be ignored.</td>
<td></td>
</tr>
</tbody>
</table>

**url**

<table>
<thead>
<tr>
<th>url: accessType inputOutput, type MFSString CDATA</th>
<th>#IMPLIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address of replacement world, activated by clicking Anchor geometry.</td>
<td>Hint: jump to a world's internal viewpoint by appending viewpoint name (e.g. #ViewpointName, someOtherCoolWorld.wrl#GrandTour).</td>
</tr>
<tr>
<td>Hint: jump to a local viewpoint by only using viewpoint name (e.g. #GrandTour).</td>
<td></td>
</tr>
<tr>
<td>Hint: Strings can have multiple values, so separate each string by quote marks [&quot;<a href="http://www.url1.org">http://www.url1.org</a>&quot;, &quot;<a href="http://www.url2.org">http://www.url2.org</a>&quot;, etc.].</td>
<td></td>
</tr>
<tr>
<td>Hint: XML encoding for &quot;is &quot; (a character entity).</td>
<td></td>
</tr>
<tr>
<td><strong>Warning:</strong> strictly match directory and filename capitalization for http links!</td>
<td></td>
</tr>
<tr>
<td>Hint: can replace embedded blank(s) in url queries with %20 for each blank character.</td>
<td></td>
</tr>
<tr>
<td>Hint: pop up a new window with url value as follows: &quot;javascript:window.open('popup.html', 'popup', 'width=240,height=240',location.href='HelloWorld.wrl')&quot;</td>
<td></td>
</tr>
</tbody>
</table>

**parameter**

<table>
<thead>
<tr>
<th>parameter: accessType inputOutput, type SFSString CDATA</th>
<th>#IMPLIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passed parameter that signals web browser how to redirect url loading.</td>
<td>Hint: set parameter to target=_blank to load target url into a blank frame.</td>
</tr>
<tr>
<td>Hint: set parameter to target=frame_name to load target url into another frame.</td>
<td></td>
</tr>
<tr>
<td>Hint: Strings can have multiple values, so separate each string by quote marks. [&quot;<a href="http://www.url1.org">http://www.url1.org</a>&quot;, &quot;etc.&quot;]</td>
<td></td>
</tr>
<tr>
<td>Interchange profile hint: this field may be ignored.</td>
<td></td>
</tr>
</tbody>
</table>

**bboxCenter**

<table>
<thead>
<tr>
<th>bboxCenter: accessType initializeOnly, type SFSVec3f CDATA</th>
<th>&quot;0 0 0&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounding box center: position offset from origin of local coordinate system.</td>
<td></td>
</tr>
</tbody>
</table>

**bboxSize**

<table>
<thead>
<tr>
<th>bboxSize: accessType initializeOnly, type SFSVec3f CDATA</th>
<th>&quot;+-1 -1 -1&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounding box size: automatically calculated, can be specified as an optimization or constraint.</td>
<td></td>
</tr>
</tbody>
</table>

**containerField**

<table>
<thead>
<tr>
<th>containerField: NMTOKEN &quot;children&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

**class**

<table>
<thead>
<tr>
<th>class CDATA</th>
<th>#IMPLIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>
Billboard node

Billboard is another X3DGroupingNode
Child-content geometry faces user
  • Special effect that improves readability or visibility

axisOfRotation determines Billboard pivot point
  • Relative to local coordinate system
  • Default is axisOfRotation='0 1 0' which swivels about vertical (Y axis)
  • Rotations unpredictable when view above (on axis)
  • Define axisOfRotation='0 0 0' for circular rotation in any direction, always fully facing the user
Billboard hints and warnings

DEF, USE allowed for multiple Billboards nodes
  • Each copy should independently face user

Put Billboard as close to moving geometry as possible, nested inside a positioning Transform
  • Usually bad idea to put Transform inside Billboard

Do not put a Viewpoint under a Billboard
  • Creates a feedback loop
  • Unpredictable behavior likely to result
Billboard example

Starting at initial viewpoint and navigating with mouse or arrow keys reveals that Billboard Text remains facing the viewer, improving readability.
**Billboard**

Billboard is a Grouping node that can contain most nodes. Content faces the user, rotating about the specified axis. Set `axisOfRotation=0 0 0` to fully face the user's camera.

**Hint:** Put Billboard as close to the geometry as possible, nested inside Transform for local coordinate system.

**Hint:** don't put Viewpoint inside a Billboard.

**Hint:** insert a Shape node before adding geometry or Appearance.

<table>
<thead>
<tr>
<th>Field</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF</td>
<td>[DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes.</td>
</tr>
<tr>
<td></td>
<td><strong>Hint:</strong> descriptive DEF names improve clarity and help document a model.</td>
</tr>
<tr>
<td>USE</td>
<td>[USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring <em>all</em> other attributes and children.</td>
</tr>
<tr>
<td></td>
<td><strong>Hint:</strong> USEing other geometry (instead of duplicating nodes) can improve performance.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong> do NOT include DEF (or any other attribute values) when using a USE attribute!</td>
</tr>
<tr>
<td>axisOfRotation</td>
<td>[axisOfRotation: accessType inputOutput, type SFVec3f CDATA &quot;0 1 0&quot;]</td>
</tr>
<tr>
<td></td>
<td>axisOfRotation direction is relative to local coordinate system.</td>
</tr>
<tr>
<td></td>
<td><strong>Hint:</strong> axis 0 0 0 always faces viewer.</td>
</tr>
<tr>
<td>bboxCenter</td>
<td>[bboxCenter: accessType initializeOnly, type SFVec3f CDATA &quot;0 0 0&quot;]</td>
</tr>
<tr>
<td></td>
<td>Bounding box center: position offset from origin of local coordinate system.</td>
</tr>
<tr>
<td>bboxSize</td>
<td>[bboxSize: accessType initializeOnly, type SFVec3f CDATA &quot;-1 -1 -1&quot;]</td>
</tr>
<tr>
<td></td>
<td>Bounding box size: automatically calculated, can be specified as an optimization or constraint.</td>
</tr>
<tr>
<td>containerField</td>
<td>[containerField: NMTOKEN &quot;children&quot;]</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>class</td>
<td>[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.</td>
</tr>
</tbody>
</table>
Collision node

Defines camera-to-object collision-detection properties between child geometry and user

- \textit{enabled}='true' blocks user navigation through the geometry
- \textit{enabled}='false' allows user navigation through the geometry

Not used for object-to-object collision detection

Authors can detect when collision occurs

- SFTime outputOnly event \textit{collideTime}
- SFBool outputOnly event \textit{isActive}
Collision detection and terrain following

Terrain following depends on +Y axis being “up”

- Other coordinate systems are possible but do not match this X3D convention
- Thus datasets using other coordinates must be converted to match

WALK mode is another form of collision detection

- Viewer's camera drops until NavigationInfo avatar rests on geometry serving as the ground plane
- Step-over distance (an avatarSize parameter) governs whether user can rise over obstacles
Collision *proxy* field

Child geometry may be quite detailed, irregular
  
  • Complicating collision-detection calculations and thus slowing rendering performance

Can substitute SFNode *proxy* child as alternate
  
  • Shape containing a Box, Sphere or Cylinder can provide simplifying geometric alternative
  
  • *proxy* geometry is not rendered

```
<Collision DEF='Example' enabled='true'>
  <Shape containerField='proxy'><Cylinder/></Shape>
  <Inline url='SomeComplicatedObject.x3d'/>
</Collision>
```
Collision hints and warnings

<NavigationInfo type='"WALK" "FLY" />'

modes support camera-to-object collision detection

Only polygonal geometry can be used for collision detection

• No points or lines

• Special limitation: no Text node collisions

• Nevertheless you can achieve the same collision effects by adding a transparent Box or other shape, thus providing necessary polygons as boundaries
Collision node X3D-Edit

- initial view outside tank glass
- zoom into tank, collision off
- blocked by glass, collision on

```
<Collision enabled="false"/>
```

Set `enabled="true"` to block zooming through glass.
Collision example

Example screen shots first show the viewer being stopped by glass geometry, then the viewer passing through the tank glass for a closer view.

Collision enabled="true" or enabled="false" result in different navigation responses.
<table>
<thead>
<tr>
<th><strong>Collision</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEF</strong></td>
</tr>
<tr>
<td>[DEF ID #IMPLIED]</td>
</tr>
<tr>
<td>DEF defines a unique ID name for this node, referencing by other nodes.</td>
</tr>
<tr>
<td>Hint: descriptive DEF names improve clarity and help document a model.</td>
</tr>
<tr>
<td><strong>USE</strong></td>
</tr>
<tr>
<td>[USE IDREF #IMPLIED]</td>
</tr>
<tr>
<td>USE means reuse an already DEF-ed node ID, ignoring <em>all</em> other attributes and children.</td>
</tr>
<tr>
<td>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.</td>
</tr>
<tr>
<td><strong>Warning:</strong> do NOT include DEF (or any other attribute values) when using a USE attribute!</td>
</tr>
<tr>
<td><strong>bboxCenter</strong></td>
</tr>
<tr>
<td>[bboxCenter: accessType initializeOnly, type SFVec3f CDATA &quot;0 0 0&quot;]</td>
</tr>
<tr>
<td>Bounding box center: position offset from origin of local coordinate system.</td>
</tr>
<tr>
<td><strong>bboxSize</strong></td>
</tr>
<tr>
<td>[bboxSize: accessType initializeOnly, type SFVec3f CDATA &quot;-1 -1 -1&quot;]</td>
</tr>
<tr>
<td>Bounding box size: automatically calculated, can be specified as an optimization or constraint.</td>
</tr>
<tr>
<td><strong>enabled</strong></td>
</tr>
<tr>
<td>[enabled: accessType inputOutput, type SBool (true</td>
</tr>
<tr>
<td>Enables/disables collision detection for children and all descendants.</td>
</tr>
<tr>
<td>Hint: former name &quot;collide&quot; in VRML 97 specification.</td>
</tr>
<tr>
<td><strong>isActive</strong></td>
</tr>
<tr>
<td>[isActive: accessType outputOnly, type SBool (true</td>
</tr>
<tr>
<td>isActive true</td>
</tr>
<tr>
<td><strong>collideTime</strong></td>
</tr>
<tr>
<td>[collideTime: accessType outputOnly, type STime CDATA #FIXED &quot;&quot;]</td>
</tr>
<tr>
<td>Time of collision between camera (avatar) and geometry.</td>
</tr>
<tr>
<td><strong>containerField</strong></td>
</tr>
<tr>
<td>[containerField: NMITOKEN &quot;children&quot;]</td>
</tr>
<tr>
<td>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.</td>
</tr>
<tr>
<td><strong>class</strong></td>
</tr>
<tr>
<td>[class CDATA #IMPLIED]</td>
</tr>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>
Additional Resources
File formatting

X3D-Edit has a Netbeans capability for formatting

- *Alt-shift-F Format* acts upon highlighted text blocks, also available via right-click menu
- Warning: do not reformat embedded ECMAscript source code

X3D Canonicalization (C14N) also reformats X3D

- Performed prior to examples being placed in archive
- Can invoke in X3D-Edit using C14N button
- Preferred method for file formatting
Pretty-print HTML capabilities

*Pretty print* means to reformat nicely in HTML, usually with color coding

- facilitates reading and printing

X3D-Edit has this Netbeans feature

- File > Print to HTML

X3dToXhtml.xslt stylesheet

- Includes indices and hyperlinks to DEF/USE, ROUTEs, images, url values, prototypes, etc.
- Can be launched via XSL Transformation button
- Available via X3D-Edit > X3D > Export menu
Launching other XSLT stylesheets

Extensible Stylesheet Language for XML (XSLT) stylesheets support a variety of conversions:

- X3dToVrml97.xslt
- X3dToClassicVRML.xslt
- X3dToXhtml.xslt
- others
Chapter Summary
Chapter Summary

Users explore X3D worlds by choosing predefined viewpoints and navigating through 3D space.

- Bindable nodes, so only one is active at a time
- Viewpoint lets authors identify key camera locations
- NavigationInfo provides options for moving around

Nodes to improve user navigability, interaction:

- Anchor makes geometric shapes linkable, like HTML
- Billboard for axis-aligned geometry facing the user
- Collision permits or blocks a user's current camera view from passing through collidable geometry
Suggested exercises

Demonstrate the ability to choose viewpoints and navigate in master Kelp Forest Exhibit scene
  • Take screen snapshot image to show what you saw
Create a “guided tour” of multiple Viewpoints for navigating a scene of interest
Switch between EXAMINE, WALK and FLY navigation in one or more browsers
Demonstrate the Anchor node by linking some text to another scene or an external web page
Use Billboard for multiple Text descriptions, linked via Anchor to bind respective viewpoints
References
References 1

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

- Chapter 4, Viewing and Navigation
- http://x3dGraphics.com
- http://x3dgraphics.com/examples/X3dForWebAuthors

X3D Resources

- http://www.web3d.org/x3d/content/examples/X3dResources.html
References 2

X3D-Edit Authoring Tool
• https://savage.nps.edu/X3D-Edit

X3D Scene Authoring Hints
• http://x3dgraphics.com/examples/X3dSceneAuthoringHints.html

X3D Graphics Specification
• http://www.web3d.org/x3d/specifications
• Also available as help pages within X3D-Edit
References  3


- [http://www.wiley.com/legacy/compbooks/vrml2sbk/cover/cover.htm](http://www.wiley.com/legacy/compbooks/vrml2sbk/cover/cover.htm)
- [http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook](http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook)
- Chapter 26 – Viewpoint
Contact

Don Brutzman

brutzman@nps.edu

http://faculty.nps.edu/brutzman

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

The Computer Graphics Educational Materials Source (CGEMS) site is designed for educators
• to provide a source of refereed high-quality content
• as a service to the Computer Graphics community
• freely available, directly prepared for classroom use
• http://cgems.inesc.pt

X3D for Web Authors recognized by CGEMS! 😊
• Book materials: X3D-Edit tool, examples, slidesets
• Received jury award for Best Submission 2008

CGEMS supported by SIGGRAPH, Eurographics
Creative Commons open-source license

http://creativecommons.org/licenses/by-nc-sa/3.0
Open-source license
for X3D-Edit software and X3D example scenes

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

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

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

• Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

• Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

• Neither the names of the Naval Postgraduate School (NPS) Modeling Virtual Environments and Simulation (MOVES) Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
X3D Graphics for Web Authors

Chapter 4

Viewing and Navigation

*But the eyes, though they are no sailors, will never be satisfied with any model, however fashionable, which does not answer all the requisitions of art.*

Henry David Thoreau, 1849
Chapter Overview
Overview: Viewing and Navigation

Users explore X3D worlds by choosing predefined viewpoints and navigating through 3D space.
• Bindable nodes, so only one is active at a time
• Viewpoint lets authors identify key camera locations
• NavigationInfo provides options for moving around

Related nodes improve navigability, interaction
• Anchor makes geometric shapes linkable
• Billboard keeps child geometry facing the user
• Collision can allow or prevent a user's view from passing through geometry
Chapter 4 - Viewing and Navigation

Concepts
Viewing and navigation

It is helpful to think of X3D scenes as fixed at different locations in 3D space
- Viewpoints are like cameras, prepositioned in locations (and directions) of interest
- Users can move their current camera viewpoint further and change direction they are looking at
- This process is called navigation

Making navigation easy for users is important
- Authors provide viewpoints of interest with scenes
- Browsers enable camera rotation, pan, zoom, etc.

Difficult navigation leads to users becoming “lost in space” or, worse yet from an author's perspective, simply leaving the scene because it is incomprehensible.
Goals of viewing and navigation

- Viewing a scene from different vantage points that reveal aspects of interest, document key locations, or help to tell a story
- Navigating changes in the user’s view of a scene effectively, by moving from place to place in an intuitive manner
- Making geometric objects selectable so that users can transport to another viewpoint, launch into another scene, or receive other web content
- Taking advantage of viewpoint location for special interactive techniques, such as user-facing billboard rotations and terrain following
Bindable nodes

Bindable nodes have a special property:
only one can be active at a time
- Bindable nodes are Viewpoint, NavigationInfo,
  Background, TextureBackground, Fog
- Each implements X3DBindableNode type interface
  for consistency
- First nodes found in scene become active by default

Implemented using a stack
- Similar to spring-loaded tray of plates in cafeteria
- One (and only one) is active, on top
- One can be pulled off top, sent off to the side
- One can be pulled to top, pushing down others

The key point here about bindable nodes is that only one of each type of node can be active and bound at a given moment.

The first bindable node (of each type) that is found in a scene becomes active by default. Any bindable nodes found within an Inline scene are added to the stack, but these Inlined nodes cannot become active automatically by default.

Four types of X3DBindableNode types:
- X3DViewpointNode: Viewpoint, OrthoViewpoint, GeoViewpoint
- NavigationInfo, GeoViewpoint
- X3DBackgroundNode: Background, TextureBackground
- Fog (but not LocalFog)

Background, TextureBackground, Fog nodes are covered in Chapter 11, Lighting and Environment nodes.


Stereo glasses are still possible with a single bound Viewpoint since the left/right eye disparities are computed as offsets from that single location and orientation.
Binding example

Basic user operation is pretty simple:
  • just select the desired Viewpoint

Complex example follows, stepping through binding stack operations
  • Advanced details
  • BindingOperations.x3d
  • Animated with scripting
  • Console results found in BindingOperations.console.txt

New users please skip ahead to Nodes and Examples

Related chapter for event passing: Chapter 8, User Interactivity Nodes

Example scene, expected output:

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

X3D for Web Authors Figure 4.1. Binding node operations: set_bind events control whether bindable nodes go to the top of the stack or else pop off the stack.

TODO: improved resolution figure

Example scene, expected output:

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

Binding node operations

• **Time t₀.** The initial loading of the scene has first `<Viewpoint DEF='View1'/>` active and bound to the top of the binding stack. Other viewpoints are off the binding stack.

• If no viewpoints are provided in the scene, then the default `<Viewpoint position='0 0 10'/>` defined in the X3D Specification is used.

• **Time t₁.** When the user selects View2 from the viewpoint list, it receives a `set_bind="true"` event and goes to the top of the binding stack. View2 also issues an `isBound="true"` event, and View1 issues an `isBound="false"` event as it moves down the stack.

These steps refer to preceding diagram
Binding node operations

- **Time t2.** Similar to the previous transitions in step t1, View3 receives a `set_bind="true"` event and responds with an `isBound="true"` event, while View2 issues an `isBound="false"` event and pushes View1 further down the stack.

- **Time t3.** View3 receives a `set_bind="false"` event, triggering a corresponding `isBound="false"` event and dropping off the stack completely. Because View2 is the next node on the binding stack, it pops to the top to become the active Viewpoint node. View2 also issues an `isBound="true"` event.

These steps refer to preceding diagram
Binding node operations

• **Time t4.** The user now selects View1 from the browser’s viewpoint list, so View1 receives a `set_bind="true"` event and sends a corresponding `isBound="true"` event. View2 is no longer bound, and is pushed down the binding stack.

• **Time t5.** View2 receives a `set_bind="false"` event while on the binding stack but unbound, and as a result, it is taken completely off the binding stack.

• **Time t6.** View1 is now removed off the binding stack via a `set_bind="false"` event, leaving no other defined Viewpoint nodes on the stack.

These steps refer to preceding diagram
Binding node operations

- **Time t7.** With no Viewpoint nodes remaining on the stack to bind, default viewpoint values are used: `<Viewpoint position='0 0 10'/>`. The user then selects the previously unbound View4 from the viewpoint list.

- **Time t8.** View4 remains as the bound viewpoint with no further viewpoints remaining on the stack.

Same process for all X3D bindable node stacks:
- Viewpoint/OrthoViewpoint/GeoViewpoint, NavigationInfo/GeoViewpoint, Fog, Background/TextureBackground

These steps refer to preceding diagram
X3D Nodes and Examples
Viewpoint node

It is helpful to think of X3D scenes as being fixed solidly in 3D space, positioned and oriented exactly where placed by the scene author.

Viewing a scene is thus a matter of navigating the current user point of view through space.

Viewpoint nodes let X3D scene authors predefine locations and orientations of particular interest:

- Sometimes viewpoints are animated and moving.
- Freedom of viewpoint is exciting and engaging, also a major advantage over fixed-viewpoint video.
Viewpoint list

Viewpoint list is optional browser-provided feature that lists currently available viewpoints

- Provides description information for viewpoints
- Simplifies user selection of viewpoints
- Thus supports navigation within a scene

Viewpoints are listed in the order that they appear in the “extended scene graph”

- First includes order of definition in primary scene
- Then includes viewpoints provided by Inline and prototype instances, inserted in order within the list
- Authors need to order Viewpoints carefully so that user navigation, understanding is best supported

Viewpoint list:

- X3d Abstract Specification, 23.2.5 Viewpoint list

  http://web3d.org/x3d/specifications/ISO-IEC-19775-1.2-X3D-AbstractSpecification/Part01/components/navigation.html#ViewpointList
Viewpoint *description*

Each Viewpoint is given a *description* string to help users decide which view to select

- Clear, understandable descriptions can guide users
- Use an object's name first when many viewpoints follow, so they are more easily identified in a list
- Use whitespace instead of underscores for better readability

Viewpoints are primary user tool for navigation

- Browsers provide Viewpoint List to show and select descriptions
- So authors should always include *description*!
Viewpoint *position, orientation*

A Viewpoint node defines a specific *position* and *orientation* for looking at a 3D scene

- Similar to a “virtual camera” vantage point

Default Viewpoint *position* is (0 0 10)

- out 10 m on +Z axis, looking back towards origin

Any changes to Viewpoint *orientation* are made relative to that default direction (along -Z axis)

- Different initial direction than other orientations
- Visualize the situation and then use right-hand rule to figure out the correct *orientation* value
Viewpoint calculator

Viewpoint Calculator provided as an author-assist tool to compute Viewpoint orientation to a given point

Inputs:
- position
- goal look-at point
- twist angle about viewing axis

Output:
- orientation

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustumExample.x3d
http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewpointCalculator.png
Viewpoint *centerOfRotation, fieldOfView*

*centerOfRotation* is a local position
- User's current view rotates about this point if the bound NavigationInfo node is in EXAMINE mode
- Can be changed by a user's LOOKAT operation picking some other geometry as new center

*fieldOfView* is preferred minimum angular width
- Shorter side of horizontal width or vertical height
- Default is 45 degrees = $\pi/4$ radians = 0.785
- Larger side determined by browser aspect ratio
- Author can set width, height if within HTML page

From X3D specification for Viewpoint node:

“The *fieldOfView* field specifies a preferred minimum viewing angle from this viewpoint in radians. A small field of view roughly corresponds to a telephoto lens; a large field of view roughly corresponds to a wide-angle lens. The field of view shall be greater than zero and smaller than $\pi$. The value of *fieldOfView* represents the minimum viewing angle in any direction axis perpendicular to the view.”

“ [...] the smaller of display width or display height determines which angle equals the *fieldOfView*”

- X3d Abstract Specification, 23.4.6 Viewpoint
ViewFrustum prototype

ViewFrustum is a helpful visualization prototype

- Prototypes simplify creation of new X3D nodes

Shows near and far clipping planes that truncate the viewable area

- Depends on Viewpoint and NavigationInfo parameters

---

ViewFrustum is now available in the X3D-Edit palette for easy drag/drop addition.

Prototypes (including this one) are covered in detail in Chapter 14. Example use:

```xml
<ExternProtoDeclare appinfo='Display view frustum associated with a given pair of Viewpoint NavigationInfo nodes'
    name='ViewFrustum' url='"ViewFrustumPrototype.x3d#ViewFrustum"
    http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter14-Prototypes/ViewFrustumPrototype.x3d#ViewFrustum'>
    <field accessType='initializeOnly' name='ViewpointNode' type='SFNode'/>
    <field accessType='initializeOnly' name='NavigationInfoNode' type='SFNode'/>
    <field accessType='inputOutput' name='lineColor' type='SFColor'/>
    <field accessType='inputOutput' name='frustumColor' type='SFColor'/>
    <field accessType='inputOutput' name='transparency' type='SFFloat'/>
    <field accessType='initializeOnly' name='trace' type='SFBool'/>
</ExternProtoDeclare>

<!-- Example use -->

<ProtoInstance name='ViewFrustum'>
    <fieldValue name='ViewpointNode'>
        <Viewpoint DEF='TestViewpoint' fieldOfView='0.78'/>
    </fieldValue>
    <fieldValue name='NavigationInfoNode'>
        <NavigationInfo DEF='TestNavigationInfo' avatarSize='1 1.6 0.75' visibilityLimit='15'/>
    </fieldValue>
    <fieldValue name='lineColor' value='0.9 0.9 0.9'/>
    <fieldValue name='frustumColor' value='0.8 0.8 0.8'/>
    <fieldValue name='transparency' value='0.75'/>
</ProtoInstance>
```
Viewpoint *jump*

*jump* can be a tricky field (but is not often used)

- *jump*='true' when a Viewpoint is selected means that the current view position and orientation is modified according to NavigationInfo *transitionType*
- *jump*='true' is usual default
- *jump*='false' is an advanced technique
  - User's view doesn't appear to change when new Viewpoint is selected
  - New Viewpoint is bound, but given offsets to match prior user position and orientation (hence no jumping)
  - Example use: changing bound viewpoint when moving from one floor into an elevator, then to another floor

Example demonstrating Viewpoint *jump* field:

http://www.web3d.org/x3d/content/examples/Basic/X3dSpecification/Elevator.x3d
http://www.web3d.org/x3d/content/examples/Basic/X3dSpecification/ElevatorAnimation.png

Sequence of screen snapshots follows. Navigate towards the elevator platform by using the up-arrow key to move forward. Once you are close enough to the bottom cone, a ProximitySensor triggers the animation to start.
Viewpoint hints and warnings

Use parent Transform node(s) for complex Viewpoint orientation and position values
  • One axis of rotation at a time can work more clearly

Keyboard shortcuts are helpful
  • PageUp PageDown Home End to select Viewpoint
  • Arrow keys to examine (rotate), pan, zoom, etc. depending on current NavigationInfo mode
  • Browser may allow Viewpoint reset after navigating

Distinguish between defined Viewpoint and current navigated user-view location, direction

Remember that initial Viewpoint direction is looking down the negative Z axis, and so any rotation changes are with respect to that direction. A helpful technique is to point in that direction, then consider how the Viewpoint orientation changes from that axis.

Try changing Viewpoint position and orientation to change direction of view. Then also try modifying a parent Transform translation and rotation values. Examples:
  • <Viewpoint position="2 1 0" orientation="1 0 0 0.1"/>
  • <Transform rotation="0 1 0 0.2"><Viewpoint position="2 1 0"/></Transform>

Note X3D-Edit feature (found in right-click context menu) that lets you wrap a new parent node around the selected node.
http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter04-ViewingNavigation/BindingOperations.x3d
http://www.x3dbook.com/examples/X3dForWebAuthors/KelpForestExhibit
then select Kelp Forest Main
TODO: better image resolution
<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Viewpoint provides a specific location and direction where the user may view the scene. Background, Fog, NavigationInfo, TexturesBackground and Viewpoint are bindable nodes.</th>
</tr>
</thead>
</table>
| def       | DEF ID (IMPLIED)  
DEF defines a unique ID name for this node, referencable by other nodes.  
Hist: descriptive DEF names improve clarity and help document a model. |
| URI       | USE ID (IMPLIED)  
USE means enter an already DEF-ed node ID, ignoring all other attributes and children.  
Hist: URLing 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 intInitOnly, type SFFloat |
|          | Text description or navigation link to be displayed for this Viewpoint.  
Hist: use spaces, make descriptions clear and readable. |
|          | Warning: without description, Viewpoint is unlikely to appear on browser Viewpoints menu. Hist: many XML tools substitute XML character references automatically if needed (like &lt;30; for it or &lt;lt; for &lt; ).  
Hist: noChange profile hint: this field may be ignored. |
| position  | position | accessType inputOutput, type SFRotation DATA "0 0 0 1.0" |
|          | Position (x, y, z, rotation) relative to local coordinate system.  
Hist: this is positionChange frame default direction (0 0 0)  
Hist: complex rotations can be accomplished axis-by-axis using parent Transform. |
| orientation | orientation | accessType inputOutput, type SFRotation DATA "0 0 1.0" |
|          | Rotation (x, y, z in radians) of Viewpoint, relative to default -2 axis in local coordinate system.  
Hist: this is orientationChange frame default direction (0 0 1)  
Hist: complex rotations can be accomplished axis-by-axis using parent Transform. |
| highlight | highlight | accessType inputOutput, type SFFloat DATA "[0.585972 0.0 .0 ]" |
|          | Preferred orientation (viewing angle) from this viewpoint to window. Small field of view roughly corresponds to a telephoto lens, large field of view roughly corresponds to a wide-angle lens.  
Hist: modifying view distance to object may be better for zooming.  
Warning: fieldOfView may not be correct for different window sizes and aspect ratios.  
Hist: noChange profile hint: this field may be ignored. |
| jump      | jump | accessType inputOutput, type SFFloat true/false "true" |
|          | Transition instantly by jumping, or smoothly animate to this Viewpoint.  
Hist: set jumpTo true for smooth camera motion when going to the viewpoint. |
| centerOfRotation | centerOfRotation | accessType inputOutput, type SFRotation DATA "0 0 0 1.0" |
|          | CenterOfRotation point relates to NavigationInfo EXAMINE node.  
Hist: centerOfRotation | centerOfRotation point relates to NavigationInfo EXAMINE node. |
| sendEvent | sendEvent | accessType outputOnly, type SFFloat (true/false) "true" |
|          | Sending event set than true makes this node active. Sending event set than false makes this inactive. Thus setting set than true false will pop/push (enable/disable) this viewpoint.  
Hist: sendEvent | sendEvent event set than true makes this node active. Sending event set than false makes this node inactive. Thus setting set than true false will pop push (enable disable) this viewpoint. |
| startStop | startStop | accessType outputOnly, type SFFloat DATA "FIXED " |
|          | Event sent when node becomes active/inactive. |
| isLoaded | isLoaded | accessType outputOnly, type SFFloat (true/false) "true" |
|          | Event sent when node becomes active. Event fired only when loaded by another node. |
| containerField | containerField | SDOMOLEKON "children" |
|          | containerField to the field label prefix indicating relationship to parent node. Examples: geometryBox, childers 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#Viewpoint](http://www.web3d.org/x3d/content/X3dTooltips.html#Viewpoint)
OrthoViewpoint

OrthoViewpoint provides an orthographic perspective-free view of a scene from a specific location and direction

- **fieldOfView** minX, maxX, minY, maxY values (default -1 -1, 1 1) define minimum and maximum extents of view, provided in units of local coordinate system
- For a rectangular display:
  \[
  \text{width/height} = (\text{maxX-minX})/(\text{maxY-minY})
  \]
- `<component name='Navigation' level='3'/>`
- Caution: often disorienting, special uses only

All other OrthoViewpoint fields are identical to Viewpoint.

OrthoViewpoint is part of the same bindable stack as Viewpoint and GeoViewpoint nodes. Only one can be bound at a single time.

Orthographic views are often used in Computer Aided Design (CAD) plans. However they can disorient a user and make a scene look quite strange, so it is usually best to avoid including them in most scenes.
Chapter 4 - Viewing and Navigation

Navigation model 1

Users can select predefined Viewpoints
  • Defines both position and direction of view

Users can further navigate around scene
  • Using pointing device or hot keys
  • Chosen viewpoint remains bound
  • Browser applies offsets using user-driven changes

These are the default navigation key responses

Figure 4.9. Recommended Keyboard Navigation Keys and Responses

Browsers are allowed to offer variations in default key navigation. This is specified in X3D Abstract Specification Annex G: Recommended Navigation Behaviours.

Excerpt:

This annex describes basic X3D scene navigation recommended practice. This recommended practice describes a browser-independent standardized keyboard interface which implements X3D frequently used scene interactivity. Features that imply interactivity are fundamental in X3D. The author expects to be able to specify multiple viewpoints in a predictable sequence, the ability to point and select, and to enable continuous navigation within the scene. Likewise the interactor expects to be able to exercise scene functionality using predictable methods.

This recommended practice is intended to allow use of a core subset of the functionality of an X3D browser, not unnecessarily limit interactive functionality which may be provided by a browser.
Navigation model 2

User's current view can itself be animated
  • ROUTE new position/direction event values to the Viewpoint itself, or to parent Transform nodes
  • User navigation offsets to that view remain in effect
  • Thus “over the shoulder” viewpoints can follow a moving object around, while still allowing user to look around while in that moving viewpoint

Lefty and Lucy shark in the Kelp Forest Main scene use this technique as virtual tour guides

Animation techniques are covered in Chapter 7, Event Animation and Interpolation.
**NavigationInfo node**

NavigationInfo indicates how a browser might best support user navigation in the scene

**Multiple NavigationInfo nodes may exist in scene**
- Or in multiple Inline scenes loaded together

NavigationInfo is an X3DBindableNode
- So only one NavigationInfo can be active at a time
- Follows the same binding rules as Viewpoint, but note that they are easily selectable by end users
- Can be linked to a given Viewpoint by ROUTE that connects `isBound` of one node to `set_bind` of other

ROUTE and animation techniques are covered in Chapter 7, Event Animation and Interpolation.
NavigationInfo *type*

Primary field is *type* which indicates which of the various modes of navigation are relevant

- "EXAMINE" best for rotating solitary objects
- "FLY" allows zooming in, out and around
- "WALK" also allows exploration, but on the ground
- "LOOKAT" use pointer to select geometry of interest
- "ANY" lets user select any mode
- "NONE" gives user zero control of navigation

MFString array default *type*=' "EXAMINE" "ANY" ' which gives users plenty of flexibility

The first supported value in the *type* list is used once a NavigationInfo values. Other specified values are usually offered by the navigation interfaces of most browsers. Any navigation modes that are not listed in the *type* list are illegal for the current NavigationInfo node and should not be offered as user-selectable options.

LOOKAT functionality is similar to user zooming in/out. *fieldOfView* does not change during this operation.

Note that browsers are also allowed to define their own specialty navigation types. In general, extensibility is good! However, there is little guarantee that such a mode will be available in other browsers. So, if you use an additional specialty *type* mode, be sure to also give the user another well-specified mode in the *type* list so that they can navigate satisfactorily.

Sometimes this flexibility lets authors and developers innovate as a group to develop good new practices. For example, given that it is a common browser feature, will we ever formalize a NavigationInfo *type* called “SHOWALL” or “FIT” in the X3D specification?
NavigationInfo type details

- **"EXAMINE"** Used to view individual objects. Scene navigation consists of rotating the user viewpoint about the center of the observed object. The `centerOfRotation` field of the currently bound Viewpoint node values determines which local point centers the view rotation.

- **"WALK"** Used when exploring a virtual world on the ground. The user’s eye level stays above the ground geometry and collision detection prevents the user from falling if underlying geometry is present.

Default values for the `type` list are “EXAMINE” “ANY” which means that a NavigationInfo node will start in EXAMINE mode, but any other mode can be selected instead by the user at run time.
NavigationInfo \textit{type} details 2

- \textbf{"FLY"} Similar to "WALK", but terrain following and collision detection is ignored. This type of navigation has the fewest constraints. Shifts the current view and related \textit{centerOfRotation} values to track or zoom toward objects of interest to user.
- \textbf{"ANY"} Browser is allowed to provide whichever navigation type seems appropriate for the task at hand, modifying the user interface if necessary.
- \textbf{"NONE"} All navigation disabled and hidden. Navigation remains possible via animation of viewpoint fields or by binding other viewpoints (using viewpoint-list selection or Anchor node).

Navigation mode \textit{NONE} means that a user cannot force navigation even if they want to. This is sometimes a useful authoring choice, but should only be applied sparingly and carefully since it can fool a user into thinking that the X3D player is frozen.
NavigationInfo speed, headlight

*speed* determines how fast navigation occurs
- Default value 1 meter/second is usually pretty slow
- Might need to vary widely from ground to space
- Might need multiple NavigationInfo nodes matching different viewpoints (high *speed* for flying, low *speed* for walking around or examining objects)

*headlight* is whether a light is shining ahead from user’s point of view
- Otherwise one or more Light nodes is needed (covered in Chapter 11), or else world goes black
NavigationInfo transitions

transitionType determines type of path followed when transitioning between viewpoints

- "ANIMATE" browser chooses smoothing algorithm
- "LINEAR" interpolation of position, orientation
- "TELEPORT" immediate repositioning to destination

transitionTime

- initial array value used for linear, otherwise multiple values can be used by browser-specific "ANIMATE"

transitionComplete lets author know when done

- SFBool boolean event sent when move is finished

**transitionType** is an inputOutput MFString array of quoted string values

**transitionTime** is an inputOutput MFBool boolean array of time intervals

**transitionComplete** is an outputOnly SFTime value that can be used to trigger other animation or scripted behaviors.

Animation techniques are covered in Chapter 7, Event Animation and Interpolation and Chapter 9, Event Utilities and Scripting.
**avatarSize** SFVec3f array

```
<NavigationInfo avatarSize='0.25 1.6 0.75'/>
```

Note that an SFVec3f is a 3-tuple with 3 component values.

```
avatarSize[0] = 0.25m
```

Allowed collision distance

```
avatarSize[1] = 1.6m
```

Viewpoint height above floor

```
avatarSize[2] = 0.75m
```

Maximum step-over height

```
Y=0
```

Local ground level
NavigationInfo \textit{visibilityLimit}

\textit{visibilityLimit} defines the maximum range that may be rendered by the browser

- Measured from the user's point of view
- Geometry beyond that distance are not drawn
- \textit{visibilityLimit}= '0.0' means no limits are imposed

Quality thumbrule: meet following relationship

- \texttt{avatarSize\_collisionDistance / visibilityLimit} < 10,000
- Avoids floating-point roundoff error on graphics card and almost-coplanar polygon tearing/aliasing
- Exactly coplanar polygons still suffer from aliasing
http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter04-ViewingNavigation/NavigationInfo.x3d
### NavigationInfo

NavigationInfo describes the viewing model and physical characteristics of the viewer's avatar:

- For inspection of simple objects, usability often improves with type "EXAMINE" or "ANY".
- For inspection of simple objects, usability often improves with type "WALK" or "FLY" support.
- Support camera-to-object collision detection, Background, Fog, NavigationInfo, TexturedBackground and Viewpoint are mandatory nodes.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF ID</td>
<td>(Mandatory)</td>
</tr>
<tr>
<td>URI (DEF ID)</td>
<td>(Optional)</td>
</tr>
<tr>
<td>Type</td>
<td>(Mandatory)</td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
</tbody>
</table>

http://www.web3d.org/x3d/content/X3dTooltips.html#NavigationInfo
Aliasing (sometimes called tearing) occurs when the rendering engine is not able to detect which polygon is closer to the viewer. Equal floating point values (or floating-point roundoff error) that leads to coincident polygons causes this problem. Essentially pixels from each polygon are overwriting each other.

In this example, most of each unit box is defined to be directly superimposed and coincident with the other box.

Moving the viewpoint varies the aliasing pattern a lot, because the floating-point roundoff error is still occurring but with slightly different values.

Relevant joke:
• Patient: Doctor, doctor, my arm hurts when I raise it over my head like this!
• Doctor: In that case, don't raise your arm.

Moral of the story: don't do that, avoid coplanar polygons.

Antialiasing is the general name for techniques to avoid this problem, most are automatically applied by hardware.

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter04-ViewingNavigation/AliasingExample.x3d
Anchor node

Anchor is another grouping node that can contain other nodes.

Geometry rendered by contained nodes is active and can be selected by user:

- User clicking on Anchor geometry launches url link (.html .x3d mailto: etc.)
  - thus similar to HTML anchor tag `<a href="url">link</a>`
- Alternatively can select a viewpoint in the scene (similar to HTML bookmark)

Selected link can replace current X3D scene, or else launch into another browser window

HTML 4.01 Recommendation, Section 12 Links:

http://www.w3.org/TR/html401/struct/links.html#h-12.1
Anchor *description*

The *description* field provides the user with a single-string summary of what is selected when the Anchor geometry is selected, e.g.

- *description*='click door, open portal to new world'
- *description*='jump to next viewpoint...'

X3D browsers usually pop up the text description when the pointing device is over the selection geometry.

Anchor *description* is similar to Viewpoint *description*.
**url** Uniform Resource Locator

The *url* field provides either

- Address to new X3D scene, HTML page, or another Web resource, or else
- Viewpoint bookmark within the scene

MFString array provides alternate url addresses

- url addresses can be either local or online
- Point to alternate versions of same resource
- X3D browser goes sequentially through ordered list, one at a time, until one retrieval succeeds

Anchor *url* rules match those for Inline *url*. 
Anchor parameter

parameter provides additional information to browser regarding redirection of loaded result

- parameter='target=_blank' sends to new frame
- parameter='target=frame4' sends to named frame
- May be ignored if browser is solely X3D capable, rather than (for instance) a Web-browser plugin

Once again, designed to match functionality found in HTML anchor tag

HTML 4.01 Recommendation, Section 16.3 Specifying target frame information:

http://www.w3.org/TR/html401/present/frames.html#adef-target
Anchor hints and warnings

Strictly match capitalization of directories and file names

• Unix and http are case sensitive and fail otherwise
• Windows is forgiving but actually this hides errors

XML escape characters

• & (ampersand) &amp;
• ' (apostrophe) &apos;
• " (double quote) &quot;

HTML 4.01 Recommendation, Section 24 Character entity references in HTML 4:

http://www.w3.org/TR/html401/sgml/entities.html
AnchorComparison.x3d scene

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

http://www.montereybayaquarium.org/efc/kelp.asp

Viewpoint link: #AngledView

"http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter03-Grouping/CoordinateAxes.x3d"
Anchor example scene, editor

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter04-ViewingNavigation/Anchor.x3d
Anchor.x3d scene, X3D-Edit

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter04-ViewingNavigation/Anchor.x3d
Anchor

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<a href="http://www.web3d.org/x3d/content/X3dTooltips.html#Anchor">

Anchor
</a>

DEPRECATED

DEPRECATED: obsolete use of X3D-IMPL. X3D-IMPL is undergoing redefinition. New Anchor API is to be introduced in new X3D releases.

NAME: X3D-IMPL

X3D authoring environments will not support these features.

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

DESCRIPTION:

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>

Anchor is a Grouping node that can contain nested nodes. Clicking Anchor's geometry loads content specified by the url field. Loaded content completely replaces current content, if parameter is in scene window.

<Anchor url="http://www.example.com">
  <Separator />
  <Separator />
  <Separator />
</Anchor>
Billboard node

Billboard is another X3DGroupingNode
Child-content geometry faces user
  • Special effect that improves readability or visibility

axisOfRotation determines Billboard pivot point
  • Relative to local coordinate system
  • Default is axisOfRotation='0 1 0' which swivels about vertical (Y axis)
  • Rotations unpredictable when view above (on axis)
  • Define axisOfRotation='0 0 0' for circular rotation in any direction, always fully facing the user

Unlike most other X3D fields, in this case a 0 0 0 vector value is allowed and given a special meaning.
Billboard hints and warnings

DEF, USE allowed for multiple Billboards nodes
  • Each copy should independently face user

Put Billboard as close to moving geometry as possible, nested inside a positioning Transform
  • Usually bad idea to put Transform inside Billboard

Do not put a Viewpoint under a Billboard
  • Creates a feedback loop
  • Unpredictable behavior likely to result

Put Transform nodes outside of the Billboard to move shapes where they need to be, then insert the Billboard just above the rotating object so that it stays in place while rotating to face the user's current view.

- Transform
  - Billboard
    - Shape
http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter04-ViewingNavigation/Billboard.x3d
Billboard example

Starting at initial viewpoint and navigating with mouse or arrow keys reveals that Billboard Text remains facing the viewer, improving readability

http://www.x3dbook.com/examples/X3dForWebAuthors/KelpForestExhibit/KelpForestMain.x3d
Billboard

<table>
<thead>
<tr>
<th>Billboard is a Grouping node that can contain more nodes. Content faces the user, rotating about the specified axis. Set autoOrientation=0 0 0 to fully face the user’s camera.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint: Put Billboard as close to the geometry as possible, nested inside Transform for local coordinate system.</td>
</tr>
<tr>
<td>Hint: don’t put Viewport inside a Billboard.</td>
</tr>
<tr>
<td>Hint: insert a Shape node before adding geometry to Appearance.</td>
</tr>
</tbody>
</table>

DEF DEF (IMPLIED)
DEF defines range ID name for this node; referenceable by other nodes.
Hint: descriptor DEF cannot improve clarity and help document a model.

USE USE (IMPLIED)
USE names reuse an already DEFed node ID, ignoring all other attributes and children.
Hint: Using other geometry (instead of duplicating nodes) can improve performance.
Warning: do NOT include DEF (or any other attribute values) when using a USE attribute.

autoorientation: accessibilityType inputOutput, type String "0 0 0" |
autoRotation: direction is relative to local coordinate system.
Hint: rot.0.0.0 always there, even.

boxCenter: accessibilityType initialOnly, type String "0 0 0" |
Bounding box center position offset from origin of local coordinate system.

boxSize: accessibilityType initialOnly, type String "0 0 0" |
Bounding box size automatically calculated, can be specified in an optimization or constraint.
Collision node

Defines camera-to-object collision-detection properties between child geometry and user

- \texttt{enabled}='true' blocks user navigation through the geometry
- \texttt{enabled}='false' allows user navigation through the geometry

Not used for object-to-object collision detection

Authors can detect when collision occurs

- SFTime outputOnly event \texttt{collideTime}
- SFBool outputOnly event \texttt{isActive}

Advanced object-to-object collision techniques are also possible using the RigidBodyPhysics component defined in X3D version 3.2.
Collision detection and terrain following

Terrain following depends on +Y axis being “up”
  • Other coordinate systems are possible but do not match this X3D convention
  • Thus datasets using other coordinates must be converted to match

WALK mode is another form of collision detection
  • Viewer's camera drops until NavigationInfo avatar rests on geometry serving as the ground plane
  • Step-over distance (an avatarSize parameter) governs whether user can rise over obstacles
Collision proxy field

Child geometry may be quite detailed, irregular
  • Complicating collision-detection calculations and thus slowing rendering performance

Can substitute SFNode proxy child as alternate
  • Shape containing a Box, Sphere or Cylinder can provide simplifying geometric alternative
  • proxy geometry is not rendered

```xml
<Collision DEF='Example' enabled='true'>
  <Shape containerField='proxy'><Cylinder/></Shape>
  <Inline url='SomeComplicatedObject.x3d'/>
</Collision>
```

Note that proxy child can occur in any order as part of the Collision node's children. The following example also includes the optional default-valued attributes for containerField='children'.

```xml
<Collision DEF='Example' enabled='true'>
  <Group containerField='children' DEF='SomeComplicatedGeometry'/>
  <Shape containerField='proxy'>
    <Cylinder/>
  </Shape>
  <Inline containerField='children' url='AnotherComplicatedObject.x3d'/>
</Collision>
```

However, note that if reformatted in X3D Canonical Form as a preparation step for X3D Compressed Binary Encoding (CBE), children nodes are sorted in order to group alike containerField values together. Additionally any default containerField values would be omitted. Thus the above example becomes:

```xml
<Collision DEF='Example' enabled='true'>
  <Shape containerField='proxy'>
    <Cylinder/>
  </Shape>
  <Group DEF='SomeComplicatedGeometry'/>
  <Inline url='AnotherComplicatedObject.x3d'/>
</Collision>
```

4.2.3 X3D canonical form
http://www.web3d.org/x3d/specifications/ISO-IEC-FCD-19776-3.2-X3DEncodings-CompressedBinary/Part03/concepts.html#X3DCanonicalForm
Collision hints and warnings

<NavigationInfo type='"WALK" "FLY" '/>

modes support camera-to-object collision detection

Only polygonal geometry can be used for collision detection

• No points or lines
• Special limitation: no Text node collisions
• Nevertheless you can achieve the same collision effects by adding a transparent Box or other shape, thus providing necessary polygons as boundaries
Collision node X3D-Edit

<table>
<thead>
<tr>
<th>Scene</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial view outside tank</td>
<td>glass</td>
</tr>
<tr>
<td>zoom into tank, collision</td>
<td>off</td>
</tr>
<tr>
<td>blocked by glass, collision</td>
<td>on</td>
</tr>
</tbody>
</table>

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter04-ViewingNavigation/Collision.x3d
Collision example

Example screen shots first show the viewer being stopped by glass geometry, then the viewer passing through the tank glass for a closer view. Collision enabled="true" or enabled="false" result in different navigation responses.

X3D for Web Authors, Figure 4.5
### Collision

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Collision detects camera-to-object contact using current Viewpoint and NavigationInfo cameraInfo. Collision is a Grouping node that handles collision detection for its children. Collision can contain a single proxy child node for primitive collision detection geometry. Note: proxy geometry is not rendered. Note: PointSet, IndexedLineSet, LineSet and Text do not trigger collisions.</td>
</tr>
<tr>
<td>DEF</td>
<td>DEF defines a unique ID name for this node, referencable by other nodes.</td>
</tr>
<tr>
<td>USE</td>
<td>USE inherits the defined DEF of node ID, ignoring all other attributes and children.</td>
</tr>
<tr>
<td>Matrix</td>
<td>Matrix is a 4x4 transformation matrix. Specifies the Local coordinate system (in view coordinates) of the node. This affects how the node (and all children) move with changes to the camera's position or orientation. Matrix is specified via an XML attribute or an inline 4x4 matrix.</td>
</tr>
<tr>
<td>time</td>
<td>Time of collision between camera (trans) and geometry.</td>
</tr>
<tr>
<td>containedField</td>
<td>containedField is the field label prefix indicating relationship to parent node. Examples: geometryBox, childrenGroup, proxyShape. containedField attribute is only supported in XML encoding of X3D scene.</td>
</tr>
</tbody>
</table>

---

http://www.web3d.org/x3d/content/X3dTooltips.html#Collision
Additional Resources
File formatting

X3D-Edit has a Netbeans capability for formatting

- *Alt-shift-F Format* acts upon highlighted text blocks, also available via right-click menu
- Warning: do not reformat embedded ECMAScript source code

X3D Canonicalization (C14N) also reformats X3D

- Performed prior to examples being placed in archive
- Can invoke in X3D-Edit using C14N button
- Preferred method for file formatting

Source Editing in Netbeans 6.0+
http://www.netbeans.org/kb/60/java/editor-tips.html

X3D Canonical Form
http://www.web3d.org/x3d/specifications/ISO-IEC-19776-3-X3DEncodings-CompressedBinary/Part03/concepts.html#X3DCanonicalForm

Issue-tracker entry for C14N addition to X3D-Edit
https://www.movesinstitute.org/bugzilla/show_bug.cgi?id=1461

TODO: add C14N to X3D-Edit Tools menu
Pretty-print HTML capabilities

*Pretty print* means to reformat nicely in HTML, usually with color coding
- facilitates reading and printing

X3D-Edit has this Netbeans feature
- File > Print to HTML

X3dToXhtml.xslt stylesheet
- Includes indices and hyperlinks to DEF/USE, ROUTEs, images, url values, prototypes, etc.
- Can be launched via XSL Transformation button
- Available via X3D-Edit > X3D > Export menu

---

Tagset pretty-printing in XHTML (.html encoding), includes cross linking of DEF/USE/ROUTE and other features: X3dToXhtml.xslt and X3dToXhtml.bat

http://www.web3d.org/x3d/content/examples/help.html#Conversions

Issue-tracker entry for X3dToXhtml.xslt addition to X3D-Edit

https://www.movesinstitute.org/bugzilla/show_bug.cgi?id=1549
Example use is shown of Export as Annotated XHTML, which invokes X3dToXhtml.xslt stylesheet to make pretty-print HTML output:
Launching other XSLT stylesheets

Extensible Stylesheet Language for XML (XSLT) stylesheets support a variety of conversions

- X3dToVrml97.xslt
- X3dToClassicVRML.xslt
- X3dToXhtml.xslt
- others

Template for new stylesheet also available: File > New X3D > New XSLT stylesheet...
Under Help system, see XML Documents > Creating and Editing an XSL Stylesheet, Performing an XSL Transformation
Chapter Summary
Chapter Summary

Users explore X3D worlds by choosing predefined viewpoints and navigating through 3D space.

- Bindable nodes, so only one is active at a time
- Viewpoint lets authors identify key camera locations
- NavigationInfo provides options for moving around

Nodes to improve user navigability, interaction:

- Anchor makes geometric shapes linkable, like HTML
- Billboard for axis-aligned geometry facing the user
- Collision permits or blocks a user's current camera view from passing through collidable geometry
Suggested exercises

Demonstrate the ability to choose viewpoints and navigate in master Kelp Forest Exhibit scene
  • Take screen snapshot image to show what you saw
Create a “guided tour” of multiple Viewpoints for navigating a scene of interest
Switch between EXAMINE, WALK and FLY navigation in one or more browsers
Demonstrate the Anchor node by linking some text to another scene or an external web page
Use Billboard for multiple Text descriptions, linked via Anchor to bind respective viewpoints
References
References

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

- Chapter 4, Viewing and Navigation
- http://x3dGraphics.com
- http://x3dgraphics.com/examples/X3dForWebAuthors

X3D Resources
- http://www.web3d.org/x3d/content/examples/X3dResources.html
References

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


- http://www.wiley.com/legacy/compbooks/vrml2sbk/cover/cover.htm
- http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook
- Chapter 26 – Viewpoint
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)
Copyright-Noncommercial-Share Alike 3.0 Unported
You are free:

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

Under the following conditions:

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

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

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

* Noncommercial. You may not use this work for commercial purposes.

* Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

* For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

* Any of the above conditions can be waived if you get permission from the copyright holder.

* Nothing in this license impairs or restricts the author's moral rights.
Open-source license
for X3D-Edit software and X3D example scenes

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

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

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

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

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

License available at

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

Good references on open source:
