X3D
- Transformations, Grouping, Navigation -

Felix G. Hamza-Lup, Ph.D
Associate Professor / Director NEWS Lab
Computer Science and Information Technology
Armstrong State University
Savannah, Georgia, USA
Outline

• 2D Transformations
  • Basic 2D transformations
  • Matrix representation

• 3D Transformations
  • Basic 3D transformations

• Transformations in X3D
  • Examples

• Grouping in X3D
  • Examples

• Navigation in X3D
Transformations

• Applied on the polygons to change their position and orientation respective to the viewpoint

• Allows definitions of objects in own coordinate systems

• Allows use of object definition multiple times in a scene
2D Transformations

• Translation:
  • $x' = x + tx$
  • $y' = y + ty$

• Scale:
  • $x' = x + sx$
  • $y' = y + sy$

• Shear:
  • $x' = x + hx*y$
  • $y' = y + hy*x$

• Rotation:
  • $x' = x*\cos\Theta - y*\sin\Theta$
  • $y' = x*\sin\Theta + y*\cos\Theta$
2D Transformations

- Scale
- Rotate
- Translate

Modeling Coordinates

Scale Translate

World Coordinates

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University
2015
2D Modeling Transformations

Modeling Coordinates

Initial location at (0, 0) with x- and y-axes aligned

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University 2015
2D Modeling Transformations

Modeling Coordinates

Scale .3, .3

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University
2015
2D Modeling Transformations

Modeling Coordinates

Scale 0.3, 0.3
Rotate -90°
2D Modeling Transformations

Modeling Coordinates

Scale .3, .3
Rotate -90
Translate 5, 3

World Coordinates

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University 2015
Scaling

- **Scaling** a coordinate means multiplying each of its components by a scalar.
- **Uniform scaling** means this scalar is the same for all components:
Scaling

• *Non-uniform scaling*: different scalars per component:

  \[ X \times 2, \quad Y \times 0.5 \]

• *How can we represent this in matrix form?*
Scaling

• Scaling operation:

\[
\begin{bmatrix}
x'

y'
\end{bmatrix} = \begin{bmatrix} ax \\
b y \end{bmatrix}
\]

• Or, in matrix form:

\[
\begin{bmatrix}
x'

y'
\end{bmatrix} = \begin{bmatrix} a & 0 \\
0 & b \end{bmatrix} \begin{bmatrix} x \\
y \end{bmatrix}
\]

scaling matrix
2-D Rotation

\[ x' = x \cos(\theta) - y \sin(\theta) \]

\[ y' = x \sin(\theta) + y \cos(\theta) \]
2-D Rotation

\[ x = r \cos(\phi) \]
\[ y = r \sin(\phi) \]
\[ x' = r \cos(\phi + \theta) \]
\[ y' = r \sin(\phi + \theta) \]

**Trig Identity…**
\[ x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta) \]
\[ y' = r \sin(\phi) \sin(\theta) + r \cos(\phi) \cos(\theta) \]

**Substitute…**
\[ x' = x \cos(\theta) - y \sin(\theta) \]
\[ y' = x \sin(\theta) + y \cos(\theta) \]
2-D Rotation

• This is easy to capture in matrix form:

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \begin{bmatrix}
  \cos(\theta) & -\sin(\theta) \\
  \sin(\theta) & \cos(\theta)
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

• Even though \(\sin(\theta)\) and \(\cos(\theta)\) are nonlinear functions of \(\theta\),
  • \(x'\) is a linear combination of \(x\) and \(y\)
  • \(y'\) is a linear combination of \(x\) and \(y\)
Basic 2D Transformations

- Translation:
  \[ x' = x + t_x \]
  \[ y' = y + t_y \]

- Scale:
  \[ x' = x \cdot s_x \]
  \[ y' = y \cdot s_y \]

- Shear:
  \[ x' = x + h_x \cdot y \]
  \[ y' = y + h_y \cdot x \]

- Rotation:
  \[ x' = x \cdot \cos\Theta - y \cdot \sin\Theta \]
  \[ y' = x \cdot \sin\Theta + y \cdot \cos\Theta \]

Transformations can be combined (with simple algebra)
Basic 2D Transformations

• Translation:
  • $x' = x + t_x$
  • $y' = y + t_y$

• Scale:
  • $x' = x \times s_x$
  • $y' = y \times s_y$

• Shear:
  • $x' = x + h_x \times y$
  • $y' = y + h_y \times x$

• Rotation:
  • $x' = x \times \cos \Theta - y \times \sin \Theta$
  • $y' = x \times \sin \Theta + y \times \cos \Theta$
Basic 2D Transformations

- **Translation:**
  - $x' = x + t_x$
  - $y' = y + t_y$

- **Scale:**
  - $x' = x \times s_x$
  - $y' = y \times s_y$

- **Shear:**
  - $x' = x + h_x \times y$
  - $y' = y + h_y \times x$

- **Rotation:**
  - $x' = x \times \cos \Theta - y \times \sin \Theta$
  - $y' = x \times \sin \Theta + y \times \cos \Theta$

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University
2015
Basic 2D Transformations

- **Translation:**
  - $x' = x + t_x$
  - $y' = y + t_y$

- **Scale:**
  - $x' = x \cdot s_x$
  - $y' = y \cdot s_y$

- **Shear:**
  - $x' = x + h_x \cdot y$
  - $y' = y + h_y \cdot x$

- **Rotation:**
  - $x' = x \cdot \cos \Theta - y \cdot \sin \Theta$
  - $y' = x \cdot \sin \Theta + y \cdot \cos \Theta$
Basic 2D Transformations

- **Translation:**
  - $x' = x + t_x$
  - $y' = y + t_y$

- **Scale:**
  - $x' = x * s_x$
  - $y' = y * s_y$

- **Shear:**
  - $x' = x + h_x * y$
  - $y' = y + h_y * x$

- **Rotation:**
  - $x' = x * \cos \Theta - y * \sin \Theta$
  - $y' = x * \sin \Theta + y * \cos \Theta$

$x' = ((x * s_x) * \cos \Theta - (y * s_y) * \sin \Theta) + t_x$

$y' = ((x * s_x) * \sin \Theta + (y * s_y) * \cos \Theta) + t_y$
Outline

• 2D Transformations
  • Basic 2D transformations
  • Matrix representation

• 3D Transformations
  • Basic 3D transformations

• Transformations in X3D
  • Examples

• Grouping in X3D
  • Examples

• Navigation in X3D
Matrix Representation

- Represent 2D transformation by a matrix

\[
\begin{bmatrix}
  a & b \\
  c & d \\
\end{bmatrix}
\]

- Multiply matrix by column vector
  ⇔ apply transformation to point

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
a & b \\
c & d \\
\end{bmatrix} \begin{bmatrix}
x \\
y
\end{bmatrix} \quad x' = ax + by \\
y' = cx + dy
\]
Matrix Representation

• Transformations combined by multiplication

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix} = \begin{bmatrix}
    a & b \\
    c & d
\end{bmatrix} \begin{bmatrix}
    e & f \\
    g & h
\end{bmatrix} \begin{bmatrix}
    i & j \\
    k & l
\end{bmatrix} \begin{bmatrix}
    x \\
    y
\end{bmatrix}
\]

Matrices are a convenient and efficient way to represent a sequence of transformations!
2x2 Matrices

• What types of transformations can be represented with a 2x2 matrix?

2D Identity?

\[
\begin{align*}
x' &= x \\
y' &= y
\end{align*}
\]

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

2D Scale around (0,0)?

\[
\begin{align*}
x' &= s_x \ast x \\
y' &= s_y \ast y
\end{align*}
\]

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]
2x2 Matrices

• What types of transformations can be represented with a 2x2 matrix?

2D Rotate around (0,0)?

\[
x' = \cos \theta \cdot x - \sin \theta \cdot y \\
y' = \sin \theta \cdot x + \cos \theta \cdot y
\]

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} =
\begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]

2D Shear?

\[
x' = x + sh_x \cdot y \\
y' = sh_y \cdot x + y
\]

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} =
\begin{bmatrix}
1 & sh_x \\
sh_y & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University 2015
2x2 Matrices

• What types of transformations can be represented with a 2x2 matrix?

2D Mirror about Y axis?

\[ x' = -x \]
\[ y' = y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \begin{bmatrix}
  -1 & 0 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

2D Mirror over (0,0)?

\[ x' = -x \]
\[ y' = -y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \begin{bmatrix}
  -1 & 0 \\
  0 & -1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
2x2 Matrices

• What types of transformations can be represented with a 2x2 matrix?

2D Translation?

\[ x' = x + t_x \]
\[ y' = y + t_y \]

NO!

Only linear 2D transformations can be represented with a 2x2 matrix
Linear Transformations

• Linear transformations are combinations of ...
  • Scale,
  • Rotation,
  • Shear, and
  • Mirror

• Properties of linear transformations:
  • Satisfies:
    • Origin maps to origin
    • Lines map to lines
    • Parallel lines remain parallel
    • Ratios are preserved
    • Closed under composition

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix} = \begin{bmatrix}
    a & b \\
    c & d
\end{bmatrix} \begin{bmatrix}
    x \\
    y
\end{bmatrix}
\]

\[
T(s_1p_1 + s_2p_2) = s_1T(p_1) + s_2T(p_2)
\]
Homogeneous Coordinates

• Q: How can we represent translation as a 3x3 matrix?

\[ x' = x + t_x \]
\[ y' = y + t_y \]
Homogeneous Coordinates

• **Homogeneous coordinates**
  • represent coordinates in 2 dimensions with a 3-vector

• Homogeneous coordinates seem unintuitive, but they make graphics operations much easier
Homogeneous Coordinates

• Q: How can we represent translation as a 3x3 matrix?

• A: Using the rightmost column:

\[
\begin{align*}
    x' &= x + t_x \\
    y' &= y + t_y
\end{align*}
\]

\[
\text{Translation} = \begin{bmatrix}
    1 & 0 & t_x \\
    0 & 1 & t_y \\
    0 & 0 & 1
\end{bmatrix}
\]
Translation

• Example of translation

• Homogeneous Coordinates

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} = \begin{bmatrix}
  x + t_x \\
  y + t_y \\
  1
\end{bmatrix}
\]
Basic 2D Transformations

• Basic 2D transformations as 3x3 matrices

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

Translate

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
\cos \Theta & -\sin \Theta & 0 \\
\sin \Theta & \cos \Theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

Rotate

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 \\
0 & s_x & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

Scale

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
1 & sh_x & 0 \\
sh_y & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

Shear
Affine Transformations

• Affine transformations are combinations of ...
  • Linear transformations, and
  • Translations

\[
\begin{bmatrix}
  x' \\
  y' \\
  w
\end{bmatrix} =
\begin{bmatrix}
  a & b & c \\
  d & e & f \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  w
\end{bmatrix}
\]

• Properties of affine transformations:
  • **Origin does not necessarily map to origin**
  • Lines map to lines
  • Parallel lines remain parallel
  • Ratios are preserved
  • Closed under composition
Projective Transformations

• Projective transformations ...
  • Affine transformations, and
  • Projective warps

\[
\begin{bmatrix}
  x' \\
  y' \\
  w'
\end{bmatrix} = \begin{bmatrix}
  a & b & c \\
  d & e & f \\
  g & h & i
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  w
\end{bmatrix}
\]

• Properties of projective transformations:
  • Origin does not necessarily map to origin
  • Lines map to lines
  • Parallel lines do not necessarily remain parallel
  • Ratios are not preserved
  • Closed under composition
Overview

• 2D Transformations
  • Basic 2D transformations
  • Matrix representation
  • Matrix composition

• 3D Transformations
  • Basic 3D transformations
  • Same as 2D
3D Transformations

• Same idea as 2D transformations
  • Homogeneous coordinates: \((x,y,z,w)\)
  • 4x4 transformation matrices

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix} = \begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  m & n & o & p
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}
\]
Basic 3D Transformations

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

**Identity**

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = \begin{bmatrix}
s_x & 0 & 0 & 0 \\
0 & s_y & 0 & 0 \\
0 & 0 & s_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

**Scale**

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & t_x \\
0 & 1 & 0 & t_y \\
0 & 0 & 1 & t_z \\
0 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

**Translation**

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = \begin{bmatrix}
-1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

**Mirror about Y/Z plane**
Basic 3D Transformations

Rotate around Z axis:
\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} =
\begin{bmatrix}
\cos \Theta & -\sin \Theta & 0 & 0 \\
\sin \Theta & \cos \Theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

Rotate around Y axis:
\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} =
\begin{bmatrix}
\cos \Theta & 0 & \sin \Theta & 0 \\
0 & 1 & 0 & 0 \\
-\sin \Theta & 0 & \cos \Theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

Rotate around X axis:
\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \Theta & -\sin \Theta & 0 \\
0 & \sin \Theta & \cos \Theta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]
Outline

- 2D Transformations
  - Basic 2D transformations
  - Matrix representation
- 3D Transformations
  - Basic 3D transformations
- Transformations in X3D
  - Examples
- Grouping in X3D
  - Examples
- Navigation in X3D
Transformations in X3D

• Transform Node can:
  • Translate local origin linearly to another coordinate
  • Rotate around any axis
  • Scale uniformly or separately along X Y Z axes

Transform fields

• translation: x y z movement in meters from origin of local coordinate system
• rotation: [axis x y z]-angle rotation about origin of local coordinate system
• scale: x y z (potentially nonuniform) factor for change in object scale to make it larger or smaller
• center: origin offset prior to applying rotation
• scaleOrientation: [axis x y z]-angle rotation to apply prior to scaling
• bboxCenter, bboxSize: x y z bounding box information (if any is provided by author, optional)
Transformations Example
Each Transform is a scene subgraph
Grouping Nodes

• Grouping nodes organize objects in an X3D world
  • Group, StaticGroup collect related nodes together
  • Transform controls position, orientation and scale
  • Inline loads other X3D scenes
  • LOD (level of detail) provides different levels of geometry quality according to the user's viewpoint
  • Switch can be animated to select different children, one (or none) at a time

• Other grouping nodes are
  • Anchor, Billboard, Collision
Why Grouping?

- X3D scenes are directed acyclic graphs, made up of subgraphs with intermediate & leaf nodes.

- Grouping nodes help provide sensible structure:
  - Functionally related nodes collected together
  - Grouping nodes can contain other grouping nodes, i.e. graphs of subgraphs
  - Establish common or separate coordinate systems
  - Make it easy to label nodes or subgraphs with DEF, then reference copies of those nodes (or grouped collections of nodes) with USE.
DEF and USE

DEF names provide a label for any node
• Including child nodes making up that subgraph
• Equivalent to ID type in XML: must be unique within an X3D scene, no duplicate node labels
• Provides target for routing events
• Multiple DEFs: legal in X3D, illegal in XML, harmful

USE labels reference a DEF node
• Spelling is case sensitive, must be identical

DEF label must precede USE reference in scene
• Enables faster performance by single-pass loading
• Not detected by XML validation but still required

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University
2015
DEF naming

Names are important!

• Describe purpose and functionality
• Strongly influences how you think about a thing
• Provides explanatory documentation
• Must start with a letter, can’t use hyphens

Naming convention: CamelCaseNaming

• capitalize each individual word
• avoid abbreviations, since none are consistent and they don't help international readers
• strive for clarity, be brief but complete
Grouping Example
Outline

• 2D Transformations
  • Basic 2D transformations
  • Matrix representation

• 3D Transformations
  • Basic 3D transformations

• Transformations in X3D
  • Examples

• Grouping in X3D
  • Examples

• Viewing & Navigation in X3D
Viewing & Navigation
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  (Anchor Example)
  • Billboard keeps child geometry facing the user (Billboard Example)
  • Collision can allow or prevent a user's view from passing through geometry
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.
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 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 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
Viewpoint centrofRotation, fieldOfView

- **centrofRotation** 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
ViewFrustum

(see ViewFrustum Example)
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
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

<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>
Navigation model 2

• User's current view can be animated (*we’ll talk about this later*)

  • 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
NavigationInfo node (1)

• 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 can be active at a given time
  • Follows the same binding rules as Viewpoint, but not 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 node (2)

• 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 node (3)

- **"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 node (4)

- "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, or else world goes black