X3D
- Texture, Material Properties, Artificial Light Models -

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

• Appearance Material and Texture

• Light Models

• More Complex Geometry
  • IndexFaceSet
  • Examples
Appearance, Material, and Textures

• Appearance affects associated geometry

• Visual surface properties that interact with lights:
  • Material and TwoSidedMaterial
  • LineProperties and FillProperties

• Texture nodes wrap images onto geometry
  • ImageTexture, MovieTexture, PixelTexture and MultiTexture
  • TextureTransform, TextureCoordinate and TextureCoordinateGenerator
Motivation

• Appearance, material and texture nodes are intended to allow authors to make 3D objects look similar to objects in the real world

• Lighting is an important factor in appearance, because 3D objects reflect their virtual light
  • Appearance and lighting are computational
  • In this chapter we assume white light available, usually from default NavigationInfo headlight
  • Lighting and environment
Parent-child constraints

• Each Shape node can contain
  • Single geometry node
  • Single Appearance node

• Each Appearance node can contain
  • A single Material (or TwoSidedMaterial) node
  • FillProperties, LineProperties, TextureProperties
  • A single Texture node (image, pixel or movie)

• Each Texture node can contain
  • Single TextureTransform or TextureTransformGenerator node
Appearance node

• Each Shape contains a single geometry node along with a corresponding Appearance node

• Appearance is a container which may include
  • A single Material (or TwoSidedMaterial) node
  • Fill/Line/Texture Properties, single Texture node

• This close association makes assignment of rendering properties to geometry unambiguous
  • Repetition of values for visual consistency is easily accomplished with DEF/USE of Appearance, Material, Texture node, etc.
  • Clear naming helps, for example

<Appearance USE='FoggyGlassAppearance'/>
Material node

• Material controls how most geometry is colored, whether it is transparent or glowing, etc.

• Surface visual properties are applied equally across all polygons making up a shape

• Material properties define how geometry visually *interacts with light sources* in the scene

• Rendering results also depend on view perspective

(!) Material is an important node to master
Material node - fields

• Color, transparency and shininess fields together make up Material properties:
  
  • `diffuseColor` reflects all X3D light sources, depending on viewing angles towards each light
  
  • `ambientIntensity` is reflection multiplication factor
  
  • `emissiveColor` is glowing component, normally off, independent of reflected light
  
  • `specularColor` governs reflection highlights
  
  • `shininess` controls specular intensity
  
  • `transparency` is ability to see through an object: 1 is completely transparent, 0 is opaque
Material specifies surface material properties for associated geometry nodes. Material attributes are used by the VRML lighting equations during rendering.

Hint: Insert Shape and Appearance nodes before adding material.

**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!
```

diffuseColor

```
[diffuseColor: accessType inputOutput, type SCFloat CDATA "0.8 0.8 0.8"]
[RGB color] how much direct, angle-dependent light is reflected from all light sources.

Hint: only emissiveColor affects IndexedLineSet, LineSet and PointSet.
```

emissiveColor

```
[emissiveColor: accessType inputOutput, type SCFloat CDATA "0 0 0"]
[RGB color] how much glowing light is emitted from this object.

Hint: emissiveColors glow even when all lights are off.

Hint: reset diffuseColor from default (0.8 .8 .8) to (0 0 0) to avoid washout.

Hint: only emissiveColor affects IndexedLineSet, LineSet and PointSet.

Warning: bright emissiveColor values can wash out some textures.
```

specularColor

```
[specularColor: accessType inputOutput, type SCFloat CDATA "0 0 0"]
[RGB color] specular highlights are brightness reflections (example: shiny spots on an apple).

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

shininess

```
[shininess: accessType inputOutput, type SFDouble CDATA "0.2"]
[0..1] low values provide soft, specular glows, high values provide sharper, smaller highlights.

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

ambientIntensity

```
[ambientIntensity: accessType inputOutput, type SFDouble CDATA "0.2"]
[0..1] how much ambient omnidirectional light is reflected from all light sources.

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

transparency

```
[transparency: accessType inputOutput, type SFDouble CDATA "0"]
[0..1] how "clear" an object is: 1.0 is completely transparent, 0.0 is completely opaque.

Interchange profile hint: transparency < .5 opaque, transparency > .5 transparent.
```

classField

```
[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.
```

containerField

```
[containerField: NMTOKEN "material"]
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.
Universal Media materials library

- The Universal Media materials were originally created by SGI as part of OpenInventor in the 1990s as a convenience to authors

- Each set of materials is grouped for visual compatibility and aesthetic appeal

- Now converted and available for X3D use
  - David Rousseau converted to VRML97
  - Aaron Walsh created VRML Universal Media archive
  - Don Brutzman translated into X3D as prototypes, cut/paste field values, also embedded in X3D-Edit

http://www.web3d.org/x3d/content/examples/Basic/UniversalMediaMaterials

Universal Media Material libraries include
ArtDeco, Autumn, Glass, Metal, Neon, Rococo, SantaFe, Sheen, Silky, Spring, Summer, Tropical, Winter

http://www.web3d.org/x3d/content/examples/Basic/UniversalMediaMaterials

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University
2015
Material editor: diffuseColor
LineProperties linetype values
(parentheses indicate optional support)

<table>
<thead>
<tr>
<th>Enumeration Code</th>
<th>linetype Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Solid</td>
</tr>
<tr>
<td>2</td>
<td>Dashed</td>
</tr>
<tr>
<td>3</td>
<td>Dotted</td>
</tr>
<tr>
<td>4</td>
<td>Dashed-dotted</td>
</tr>
<tr>
<td>5</td>
<td>Dash-dot-dot</td>
</tr>
<tr>
<td>6</td>
<td>(single arrow)</td>
</tr>
<tr>
<td>7</td>
<td>(single dot)</td>
</tr>
<tr>
<td>8</td>
<td>(double arrow)</td>
</tr>
<tr>
<td>9</td>
<td>(chain line)</td>
</tr>
<tr>
<td>10</td>
<td>(center line)</td>
</tr>
<tr>
<td>11</td>
<td>(hidden line)</td>
</tr>
<tr>
<td>12</td>
<td>(phantom line)</td>
</tr>
<tr>
<td>13</td>
<td>(break line 1)</td>
</tr>
<tr>
<td>14</td>
<td>(break line 2)</td>
</tr>
<tr>
<td>15</td>
<td>User-specified dash pattern</td>
</tr>
<tr>
<td>1</td>
<td>1 Solid</td>
</tr>
<tr>
<td>2</td>
<td>2 Dashed</td>
</tr>
<tr>
<td>3</td>
<td>3 Dotted</td>
</tr>
<tr>
<td>4</td>
<td>4 Dashed-dotted</td>
</tr>
<tr>
<td>5</td>
<td>5 Dash-dot-dot</td>
</tr>
<tr>
<td>6</td>
<td>6 (single arrow)</td>
</tr>
<tr>
<td>7</td>
<td>7 (single dot)</td>
</tr>
<tr>
<td>8</td>
<td>8 (double arrow)</td>
</tr>
<tr>
<td>9 [no entry]</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10 (chain line)</td>
</tr>
<tr>
<td>11</td>
<td>11 (center line)</td>
</tr>
<tr>
<td>12</td>
<td>12 (hidden line)</td>
</tr>
<tr>
<td>13</td>
<td>13 (phantom line)</td>
</tr>
<tr>
<td>14</td>
<td>14 (break line 1)</td>
</tr>
<tr>
<td>15</td>
<td>15 (break line 2)</td>
</tr>
<tr>
<td>16</td>
<td>16 (user-specified dash pattern)</td>
</tr>
</tbody>
</table>
Texture nodes

- Texture nodes read 2D image (or movie) files and apply them pixel-by-pixel to the associated geometry sharing the same Shape node
  - Thus wrapping picture images around an object
  - ImageTexture, PixelTexture, MovieTexture
  - Can be inexpensive way to achieve high fidelity

- Texture images can be shifted, rotated, scaled
  - TextureTransform, TextureCoordinate
  - Thus modifying image application to geometry
Common fields for Texture nodes

• repeatS and repeatT
  • These boolean fields indicate whether the texture image is repeated along a given axis once used
  • Default is to use once along each axis, mapping the texture image once from coordinates (0,0) to (1,1)

• Hint: rather than working with repeatS repeatT parameters or TextureTransform, it is often easiest to adjust a texture by modifying it within an image editor. (e.g. Adobe FireWorks CS3)
ImageTexture node

• ImageTexture retrieves a 2D image file and applies it as a texture to geometry
  • Commonly used technique, important to master

• url
  • as part of Inline and Anchor
  • Recall that the url field is an ordered list which can include both local (relative) and online addresses to image files
  • Might preferentially load online version first, perhaps if it can be updated, and keep a local url value for a backup image
Outline

• Appearance Material and Texture

• Light Models

• More Complex Geometry
  • IndexFaceSet
  • Examples
Light Models in X3D
Light – a computational model

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

• Frame rate (measured in fps, frames per second) must be high for users to believe a 3D scene is interactive and responsive to their touch
  • Human perception of smooth motion is 7-8 fps
  • Keep frame rate above 10-15 fps in your scenes

• Adding multiple lights in a scene can improve realism, if done properly
  • Contributions of each can be added, pixel by pixel

• However lights add significant computational cost, doubling/tripling/etc. rendering overhead
  • Thus light management requires careful control
Approximations

• Ultra-high-fidelity computation of lighting effects is possible, and subject of ongoing research

• Nevertheless the computational complexity of such effects are prohibitive for most computers

• Thus a number of careful approximations made in lighting equations and material functions, allowing reasonable rendering to occur while supporting interactive real-time performance
Render Techniques

• A variety of higher-fidelity systems are available that use different techniques
  • Ray tracing
  • Volumetric rendering
  • Light fields, etc.

• Nevertheless most interactive 3D graphics systems use identical (or at least similar) lighting equations and rendering techniques

• Following pre-computation, advanced techniques can often export results for 3D rendering

• Thus X3D is a good match for each approach when interactive export to Web is needed
Balance: Performance vs. Realism

• Authoring the right mix of lighting, appearance and viewing is both technical and artistic.
  • This is why both types of activity are prevalent in the graphics community

• Making a scene “look good” sometimes requires technical shortcuts or workarounds that do not exactly match the original real-world effects.

• Carefully applying these techniques to achieve an attractive, functional scene is the primary challenge facing all 3D graphics authors.
Common Lighting Fields

• ambientIntensity
  • determines brightness of assumed reflected background ambient lighting, which occurs in most indoor and outdoor locations.
  • this field also helps ensure that objects remain visible.

• intensity
  • indicates the direct brightness of a light shining from the source location
  • can be used to adjust relative strengths of multiple lights

• color
  • specifies the red-green-blue (RGB) spectral components of the light source
  • adding to (or subtracting from) all three components affects the output by whitening (or darkening) the light source
Common Lighting Fields

• global
  • indicates whether the scope of a light affects the entire 3D scene, or simply the local geometry shared within a common parent grouping node.
  • this can be expensive in large scenes, do not set `global='true'` unless the effect is intentional

• on
  • is the boolean field which turns a light on/off, allowing simple animation effects
X3D Lighting Model Limitations

- Light shines through geometry and illuminates objects that block each other
  - Necessary limitation to support real-time rendering
  - Can be surprising but usually not noticeable

- No shadows
  - Might be supported in a future X3D specification

- Placing a light inside an object is problematic
  - Only illuminates back faces, resulting in no light to externally visible faces
  - Avoid, unless intentionally showing light location

- Maximum number of active lights: 8
  - Can use more if turned off/on appropriately
  - Matches limits of OpenGL, DirectX, GPU hardware
  - Actually this is a high number for most applications
X3D Light Nodes
DirectionalLight Node

• DirectionalLight provides uniform-intensity light as set of parallel rays sharing a single direction
  • Each surface facing the light gets some illumination
  • Backface polygons, surfaces parallel to light rays remain unlit by direct intensity
  • All surfaces lit by ambientIntensity, facing or not

• Angle of incidence/reflection and appearance properties of lit shapes are primary factors on reflected color, brightness of lit objects

• Emulates distant light source, such as the sun
  • Intensity does not diminish with distance
Directional Light

No Shadows, Constant intensity

(Example)
**DirectionalLight Fields**

- ambientIntensity, color, global, intensity, on – same
  - direction is x-y-z vector indicating direction of rays in local coordinate system, which in turn depends on rotations in parent Transform node hierarchy

- Hints
  - Misdirected DirectionalLight nodes hard to debug, since light nodes themselves are not visible
  - Substitute a PointLight node to get indication of where the light actually is located
  - Can rotate DirectionalLight via parent Transform, also semitransparent Cone for direction, range
  - Position irrelevant since constant intensity, direction
The NavigationInfo node's headlight field enables a special DirectionalLight that follows user's current view location, pointing straight ahead

- Enables a light source that is always pointing in direction of user's view, ensuring objects are lit
- headlight='true' by default, color is always white

- Headlight matches the following characteristics, pointed out from center of current active view:

  <DirectionalLight color='1 1 1' intensity='1' ambientIntensity='1' direction='0 0 -1'/>

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

- PointLight emulates single light source that radiates equally in all directions
  - Analogy: single small light

- PointLight intensity varies with distance to object
  - Longer travel means less light arrives from source

- Rotation is irrelevant since light is omnidirectional

- Shared traits with other lights:
  - No shadows, light not blocked by other geometry
  - No visible representation in space of light itself
PointLight Fields

- ambientIntensity, color, global, intensity, on – same

- attenuation array holds constant, linear, quadratic coefficients for distance-attenuation factor
  - factor = 1 / max (atten[0] + atten[1]*r + atten[2]*r^2, 1)
  - Note that non-constant (linear and quadratic) coefficients are computationally expensive per pixel

- location is SFVec3f position indicating origin of rays
  - relative to local coordinate system, which in turn depends on translations and rotations in parent

- Transform node hierarchy
  - radius gives maximum effective range of light rays from source location, must be nonnegative
SpotLight Node

• SpotLight illuminates shapes within conical beam
  • Intensity decreases with distance from source

• Spotlight can be considered similar to PointLight with exception that computed light is constrained to be within conical beams, defined by solid angles
  • Center portion of cone (indicated by beamWidth) receives full intensity
  • Outer portion of light cone has linear drop off of reduced intensity (indicated by cutOffAngle)
SpotLight Fields

- ambientIntensity, color, global, intensity, on – same
  - attenuation array holds constant, linear, quadratic

- coefficients for distance-attenuation factor
  - factor = 1 / max (atten[0] + atten[1]*r + atten[2]*r^2, 1)
  - Note that non-constant (linear and quadratic) coefficients are computationally expensive per pixel

- location is SFVec3f position indicating origin of rays relative to local coordinate system, which in turn depends on translations and rotations in parent

- Transform node hierarchy
  - radius gives maximum effective range of light rays from source location, must be nonnegative
SpotLight Fields

• beamWidth specifies the half angle (in radians)
  • about the SpotLight direction that corresponds to maximum uniform intensity

• cutOffAngle specifies the half angle (in radians)
  • about the SpotLight direction that corresponds to outer bound of SpotLight node's effect.
  • Outside of this solid angle, no light is provided.

• Constraint: $0 \leq \text{beamWidth} \leq \text{cutOffAngle} \leq \pi/2$
  • direction is x-y-z vector indicating direction of rays in local coordinate system, which in turn depends on rotations in parent Transform node hierarchy
Examples

• Please see the examples online [Here]

• Try to develop your own light system for your X3D scene.
Outline

• Appearance Material and Texture

• Light Models

• More Complex Geometry
  • IndexFaceSet
  • Examples
Triangle

• Basic Polygon
  • rigid, low level geometry
  • all points in the same plane (Euclidian Geometry Axiom)
  • complex objects can be reduced to a set of triangles (called tessellation)

• Single sided vs double sided Polygons
  • Single sided rendering twice as fast
  • Better go single sided if back is not visible to user (called backface culling)

• The “Solid” attribute in X3D:
  • set solid =‘true’ means do not draw the back-face (default)
  • set solid=‘false’ means draw both faces (see Example)
Normal to the Surface

• Normals are perpendicular vectors to the surface

• Direction of the normal is defined by the order of the points defining the polygon and the Right-Hand rule.
  • Positive Normal = front face of polygon
  • Negative Normal = back face of polygon
Right-hand rule for polygon normals

normal vector is at polygon centroid, with perpendicular direction according to right-hand rule
Remember the Polygon
IndexFaceSet

- **coordIndex:**
  - Provide arrays of XYZ coordinates that connect individual vertices into polygons
  - Initial index is 0
  - Sentinel value -1 ends polygon or polyline, next coordinate is the next polygon

- **point:**
  - array of xyz coordinates
  - Each set of 3 coordinates is a point in 3D

see Example
Recommended Exercise

• Try to develop your own IndexFaceSet with at least 20 polygons

• Search online for complex X3D objects that are defined using IndexFaceSet. Bring those objects into your X3D scene.