X3D Graphics for Web Authors

Chapter 6

Points, Lines and Polygons

Drawing is a struggle between nature and the artist, in which the better the artist understands the intentions of nature, the more easily he will triumph over it. For him it is not a question of copying, but of interpreting in a simpler and more luminous language.

Charles Baudelaire, *On the Ideal and the Model,* 1846.

Contents

Chapter Overview

Many different geometry nodes

An excellent aspect of X3D is that there are many different ways to create geometry

- Chapter 2, Geometry Primitives
- Chapter 6, Points, Lines and Polygons
- Chapter 10, Geometry2D Nodes
- Chapter 13, Triangles and Quadrilaterals

These are all handled consistently inside a Shape node with corresponding Appearance

Fundamental geometry nodes

Geometry nodes in this chapter include points, lines, and indexed face sets

These nodes fundamental and can represent almost any shape

- Tools can convert other geometry to simpler forms
- Thus most are part of Interchange profile for broadest possible usage and adaptability

Some browsers support viewing geometry in wireframe (line) or point (cloud) mode, which can help to reveal internal geometric structure

- http://en.wikipedia.org/wiki/Wire-frame_model

Overview: Points, Lines and Polygons

Triangles, single-sided polygons, normal vectors

Common fields: *ccw*, *convex*, *creaseAngle*, etc.

Geometry nodes, part 2:

- Coordinate and CoordinateDouble
- Color and ColorRGBA
- PointSet
- IndexedLineSet and LineSet
- IndexedFaceSet
- ElevationGrid
- Extrusion

Concepts

Triangles

Triangles are the primary low-level geometry construct used by graphics software, hardware

- More complex shapes are reduced to triangles by the rendering software (known as tesselation)
- A triangle is always planar, allowing the material appearance to fill it

Sometimes quadrilaterals are used, but problem is that values might be non-coplanar due to roundoff (or authoring) error

- Which means that filling in material is ill defined, and not properly or repeatably renderable

Single-sided polygons

Graphics engines always prefer simplicity in order to achieve maximum run-time performance

- Top 3 considerations for graphics hardware: performance, performance, performance!

Single-sided polygons take about half the time to draw than double-sided polygons

- So if authors can arrange geometry so that only one side is ever visible to user, can go single-sided
- Technical term: *backface culling*
- Efficiency is rationale for many X3D default values
- Example: default setting is *solid*='true'
- Debugging hint: set *solid*='false' to show both sides

Review from Chapter 2

Common field: *solid*

In 3D graphics, all triangles have 2 sides

- Graphics term: backface culling only draws front sides

The *solid* field defines whether a geometry node has an inside or not, with a default value of true

- *solid*='true' means do not render (draw) the inside
- *solid*='false' means render both inside and outside

This approach reduces the number of polygons needing to be drawn, thus improving performance

Confusing if user gets lost inside invisible geometry

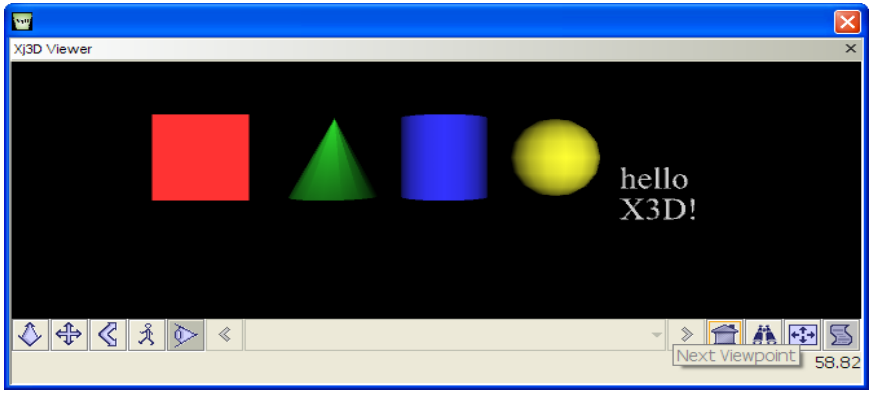
- Hint: set *solid*='false' to draw both sides

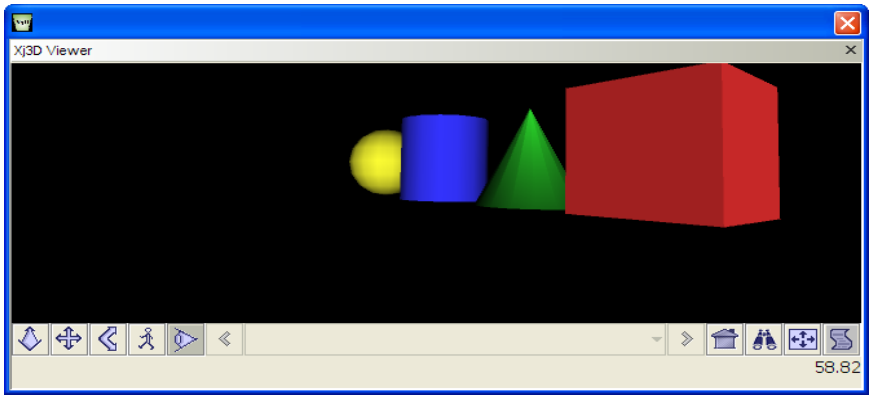
Review from Chapter 2

Common field: *solid*

To see an example of 'solid' geometry, rotate the GeometryPrimitives.x3d scene by 180 degrees

- Once rotated, the first four shapes remain visible, but the Text node disappears
- This is because *solid='true'* by default, so the reverse side of text is not drawn by default





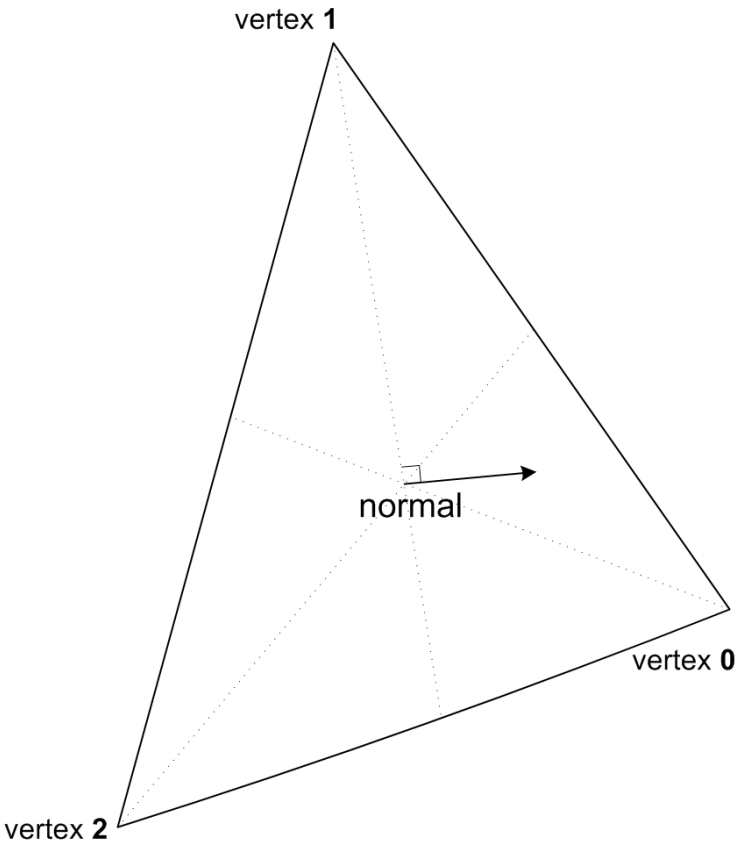
Normal vectors

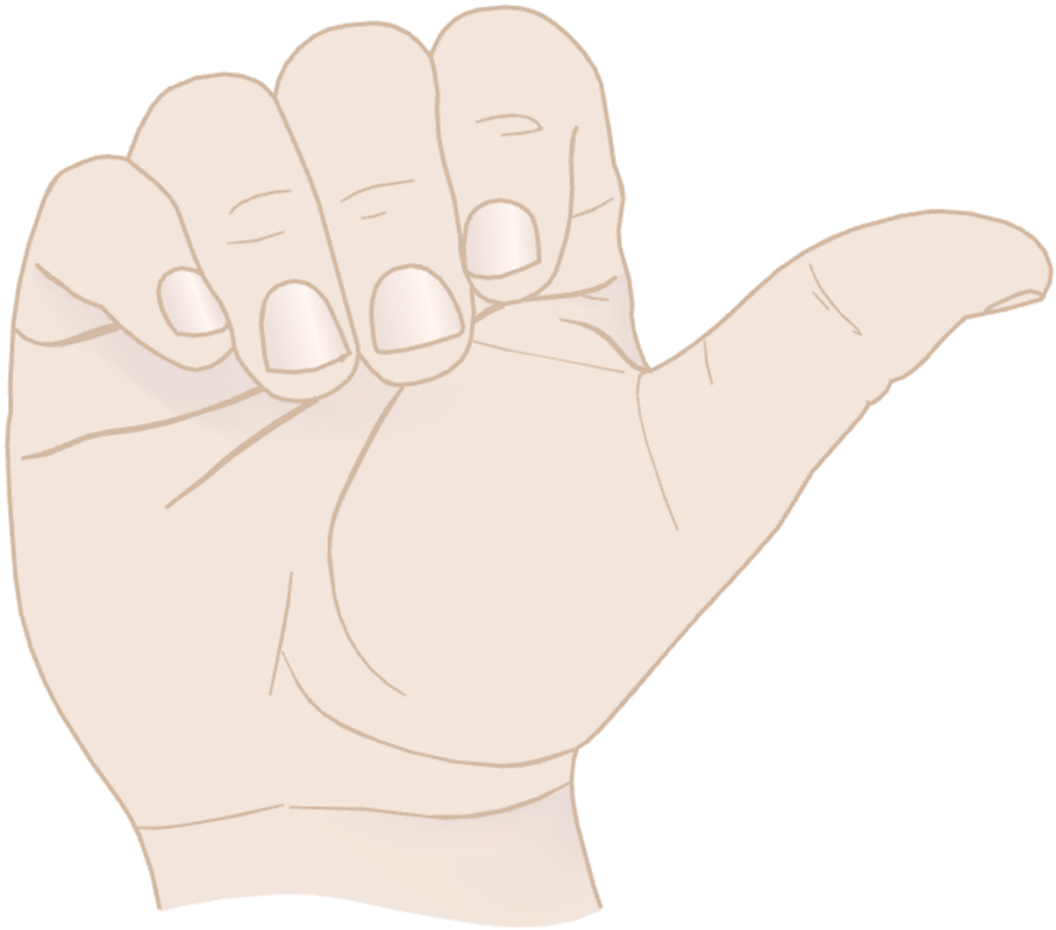
The *normal* vector is perpendicular to the face, pointing away from the centroid of polygon

Direction of normal vector defined by order of points defining the polygon and right-hand rule

- Align curvature of fingers to match polygon vertex points in order: indices 0, 1, 2 ...
- Thumb points in direction of (positive) normal, which is the front-facing side
- Negative normal thus points in direction of backface

Right-hand rule for polygon normals





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

Common field: *ccw*

ccw (counter clockwise) indicates whether default direction of polygon normals is counterclockwise (default) or clockwise

- *ccw*='true' is right-hand rule
- *ccw*='false' is opposite

Hint: can correct some opposite-rendering geometry by reversing ccw value, rather than reordering all coordinates or indices

- Saves time on some import conversions

Common geometry node patterns

```
<IndexedFaceSet>
  <Coordinate/>
  <Color/>
  <Normal/>
</IndexedFaceSet>
```

```
<IndexedLineSet>
  <Coordinate/>
  <Color/>
</IndexedLineSet>
```

etc.

Common field: *colorPerVertex*

colorPerVertex indicates whether contained color values are applied to each vertex point (default), or to each polygonal face

- *colorPerVertex*='true' requires that # colors must equal # points
- *colorPerVertex*='false' requires that # colors must equal # polygons

IndexedFaceSet *colorPerVertex* true, false

IndexedLineSet *colorPerVertex* true, false

ColorPerVertexExamples.x3d - Editor

ColorPerVertexExamples.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Interchange' 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='ColorPerVertexExamples.x3d' name='title'/>
    <meta content='Geometry Polygons Nodes: Color, Coordinate, ElevationGrid, Extrusion, IndexedFaceSet, IndexedLineSet, PointSet' name='description'/>
    <meta content='Don Brutzman' name='creator'/>
    <meta content='5 September 2005' name='created'/>
    <meta content='5 September 2005' name='modified'/>
    <meta content='Copyright (c) 2005, Daly Realism and Don Brutzman' name='rights'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ColorPerVertexExamples.x3d' name='identifier'/>
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <Viewpoint description='ColorPerVertex Examples' position='0 0 6'/>
    <Background groundColor='0.1 0.1 0.1' skyColor='0.1 0.1 0.1'/>
    <Transform translation='-0.5 0 0'>
      <Transform translation='-3 0 0'>
        <Shape>
          <IndexedFaceSet colorIndex='0 1 2 3 0 -1' colorPerVertex='true' coordIndex='0 1 2 3 0 -1' solid='true'>
            <Coordinate DEF='FourPoints' point='0 0 0 1 0 0 1 1 0 0 1 0'/>
            <Color DEF='FourColors' color='1 0 0 0 1 0 0 0 1 0.8 0.8 0.8'/>
          </IndexedFaceSet>
        </Shape>
      </Transform>
      <Transform translation='-1 0 0'>
        <Shape>
          <IndexedFaceSet colorIndex='0' colorPerVertex='false' coordIndex='0 1 2 3 0 -1' solid='true'>
            <Coordinate USE='FourPoints'/>
            <Color USE='FourColors'/>
          </IndexedFaceSet>
        </Shape>
      </Transform>
      <Transform translation='1 0 0'>
        <Shape>
          <IndexedLineSet colorIndex='0 1 2 3 0 -1' colorPerVertex='true' coordIndex='0 1 2 3 0 -1'>
            <Coordinate USE='FourPoints'/>
            <Color USE='FourColors'/>
          </IndexedLineSet>
        </Shape>
      </Transform>
      <Transform translation='3 0 0'>
        <Shape>
          <IndexedLineSet colorIndex='0' colorPerVertex='false' coordIndex='0 1 2 3 0 -1'>
            <Coordinate USE='FourPoints'/>
            <Color USE='FourColors'/>
          </IndexedLineSet>
        </Shape>
      </Transform>
    </Transform>
  </Scene>
</X3D>
```

Edit Color

containerField: color

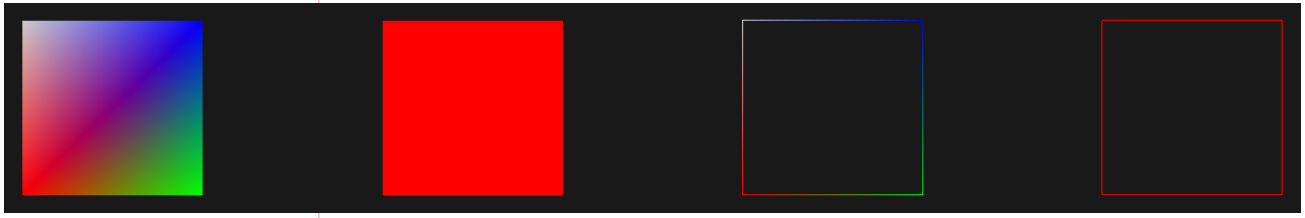
DEF: FourColors

USE:

color array

	r	g	b	color
0	1	0	0	
1	0	1	0	
2	0	0	1	
3	0.8	0.8	0.8	

OK Cancel Help

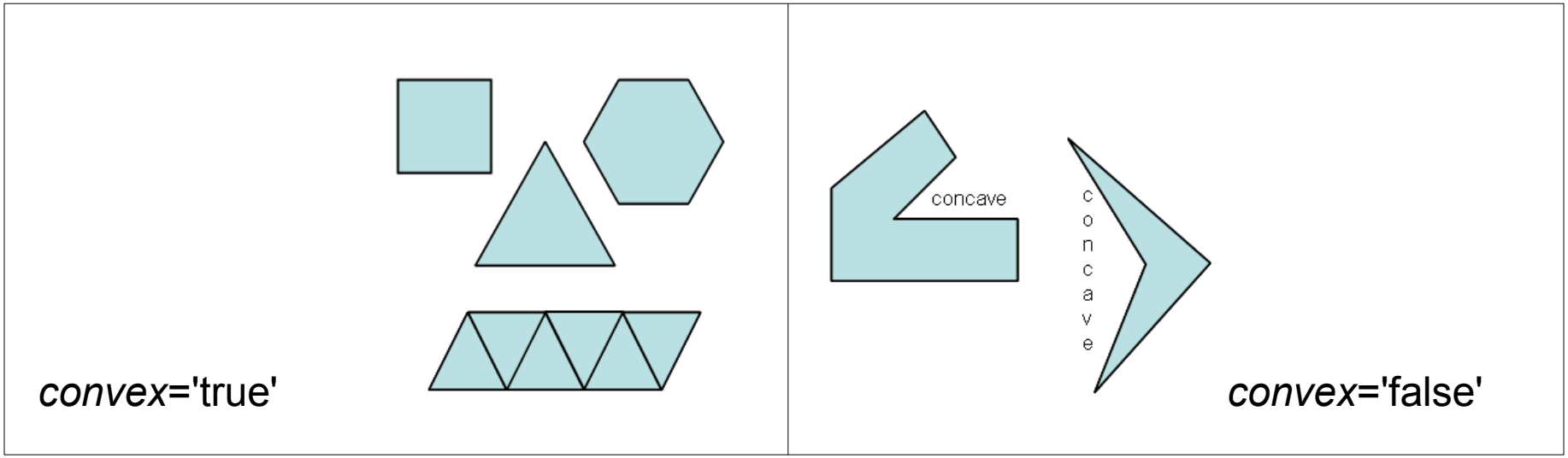


23:1 INS

Common field: *convex*

convex indicates whether an *n*-sided polygon has concave sides, meaning empty-space cavities

- *convex*='true' (default) means no concave sides
- *convex*='false' means concave sides may exist in the polygon, so extra care is needed to avoid hardware or software difficulty when rendering



Common field: *creaseAngle*

creaseAngle defines the angle (in radians) used to determine whether adjacent polygons are drawn with sharp edges or smooth shading

- If angle between polygons is less than *creaseAngle,* then smooth shading is used
- Smooth shading can conceal underlying tessellation

creaseAngle only affects shading within a single geometric shape, not exterior boundaries

- *creaseAngle*='0.0' means all edges are sharp
- *creaseAngle*='3.14159' means no edges are sharp

Common field: *normalPerVertex*

normalPerVertex indicates whether contained normal values are applied to each vertex point (default), or to each polygonal face

- *normalPerVertex*='true' requires that # normals must equal # points
- *normalPerVertex*='false' requires that # normals must equal # polygons

Common index fields: *coordIndex*, *colorIndex*, *normalIndex*

coordIndex, *colorIndex*, *normalIndex* each provide arrays of integer indices that connect individual vertices into polygons, then correlate corresponding Color/ColorRGBA, Coordinate or Normal values

- Initial index is 0
- Sentinel value -1 concludes polygon, polyline
- Maximum value equals (count - 1)
- Integer type MFInt32, default is empty array

index counting checks

- *colorIndex* count must equal (**point** count - 1) when *colorPerVertex*='true', which is default

- *colorIndex* count must equal (**polygon** count - 1) when *colorPerVertex*='false'

- *normalIndex* count must equal (**point** count - 1) when *normalPerVertex*='true', which is default

- *normalIndex* count must equal (**polygon** count - 1) when *normalPerVertex*='false'

X3D Nodes and Examples

Coordinate node

Provide array of x-y-z point values

- Required – otherwise no geometry to draw!
- Type MFVec3f array of 3-tuple values, each with 32-bit single-precision floating point

Coordinate *point* values define all of the vertices needed to build polygonal geometry

- *coordIndex* array in parent geometry node indicates connectivity for each individual polygon
- *coordIndex* value -1 indicates end of one polygon, next coordIndex value indicates vertex point that begins a new polygon

CoordinateDouble node

Definition and usage similar to Coordinate node

Provide array of x-y-z point values

- Type MFVec3d array of 3-tuple values, each with 64-bit double-precision floating point

Double precision may be needed for specialty applications (geographic, atomic, etc.)

Note however that most graphics hardware is exclusively single-point precision, for speed

- So browser may need special software techniques to handle double precision fidelity properly

Color.x3d - Editor

Color.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-inst
  <head>
    <meta content='Color.x3d' name='title'/>
    <meta content='Vertex color applied to IndexedFaceSet built positive-displacement c
                  static because the PositionInterpolators contain an initial offest c
    <meta content='Todd Gagnon and Mark A. Boyd' name='authors'/>
    <meta content='Xeena VRML importer' name='translator'/>
    <meta content='8 June 1998' name='created'/>
    <meta content='20 December 2002' name='imported'/>
    <meta content='3 February 2007' name='modified'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryP
    <meta content='KelpTank.x3d' name='reference'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/P
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html' name='generator'/
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <WorldInfo info='DTG of last update: 081200Jun98 Added: Updated: cycleInterval to 5
    <ExternProtoDeclare name='WhereAmI' url='"../Chapter14-Prototypes/WhereAmI.x3d#Wher
    <ProtoInstance name='WhereAmI'/>
    <Background skyColor='1 1 1'/>
    <Viewpoint description='Book View' orientation='0 -1 0 0.53' position='-2.28 0.29 4
    <Group>
      <Shape>
        <Appearance DEF='pumpHouse'>
          <Material diffuseColor='0.82 0.78 0.74'/>
        </Appearance>
        <IndexedFaceSet
            ccw='true'
            colorIndex='0 0 0 0 -1 0 0 1 1 -1 1 1 1 1 -1 1 0 0 1 -1 0 0 1 1 -1 1 1 1 -1 0 0 1 1 -1 0 0 1 1 -1 1 1 1 -1 0 0 0 0 -1 0 0 0 0 -1 0 0
            colorPerVertex='true'
            coordIndex='0 1 5 4 -1 5 1 2 6 -1 6 2 3 7 -1 3 0 4 7 -1 1 12 13 2 -1 2 13 14 -1 12 15 16 13 -1 15 0 3 16 -1 16 3 17 -1 9 5 6 10 -1 8
          <Coordinate DEF='CoordinateNodeExample' point='0.0 0.0 0.0 2.0 0.0 0.0 2.0 1.75 0.0 0.0 1.75 0.0 0.625 0.75 0.0 1.0 0.75 0.0 1.0 1.6 0.0
          <Color DEF='ColorNodeExample' color='.82 .78 .74 .66 .37 .02'/>
        </IndexedFaceSet>
      </Shape>
      <Transform scale='0.91 0.6 0.3' translation='0.8 -0.65 0.5'>
        <Shape>
          <Appearance>
            <Material diffuseColor='0.749 0.694 0.651'/>
          </Appearance>
          <Cylinder bottom='false' top='false'/>
        </Shape>
```

Edit Coordinate

DEF CoordinateNodeExample

USE

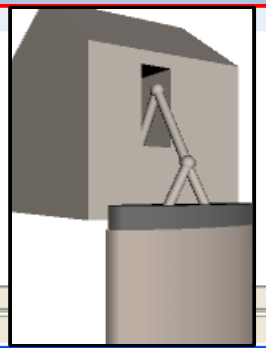
containerField coord

point array

	x	y	z
0	0	0	0
1	2	0	0
2	2	1.75	0
3	0	1.75	0
4	0.625	0.75	0
5	1	0.75	0
6	1	1.6	0
7	0.625	1.6	0
8	0.625	0.75	-0.65
9	1	0.75	-0.65
10	1	1.6	-0.65
11	0.625	1.6	-0.65
12	2	0	-2.7
13	2	1.75	-2.7
14	2	2.5	-1
15	0	0	-2.7
16	0	1.75	-2.7
17	0	2.5	-1

+ -

OK Cancel Help



37:1 INS

Coordinate	Coordinate builds geometry using a set of 3D coordinates. Coordinate is used by IndexedFaceSet, IndexedLineSet, LineSet and PointSet. Coordinate is also used by NurbsPositionInterpolator and NurbsOrientationInterpolator.
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!
point	**[point: accessType inputOutput, type MFVec3f CDATA #IMPLIED]** point contains a set of 3D coordinates.
containerField	**[containerField: NMTOKEN "coord"]** 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.

CoordinateDouble	CoordinateDouble builds geometry using a set of 3D coordinates. CoordinateDouble is used by IndexedFaceSet, IndexedLineSet, LineSet and PointSet. CoordinateDouble is also used by NurbsPositionInterpolator and NurbsOrientationInterpolator.
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!
point	**[point: accessType inputOutput, type MFVec3d CDATA #IMPLIED]** point contains a set of 3D coordinates.
containerField	**[containerField: NMTOKEN "coord"]** 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.

Color node

Color values for individual polygons, line segments and points can be defined using the Color node

Color values are red-green-blue (RGB) [0..1]

- Type is MFColor array of 3-tuple values
- HTML, SVG colors are [0..255] [#000000..#FFFFFF] and so must be converted numerically if used

Appearance and Material node can also be used to control overall transparency, if needed

- Note: Color node overrides Material color values

ColorRGBA node

ColorRGBA used similarly to Color node, but adds alpha (opacity) to each red-green-blue value

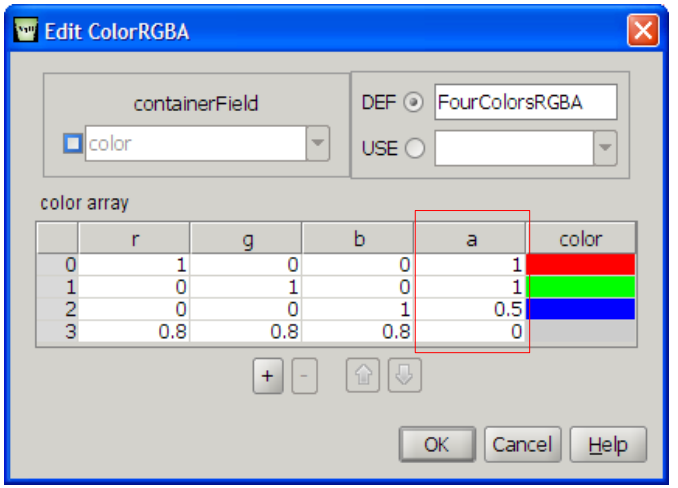
- alpha component equals (1 – transparency)

Alpha values range 0 to 1

- 0 means fully transparent
- 1 means fully opaque

RGBA values selectively allow transparent parts in geometry

- Rather than single Material transparency consistently across full geometry with Color node



Color.x3d - Editor

Color.x3d

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
3  <X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/s
4    <head>
5      <meta content='Color.x3d' name='title'/>
6      <meta content='Vertex color applied to IndexedFaceSet built positive-displacement cylinder pump house.  Note that this scene is not really
7                    static because the PositionInterpolators contain an initial offest of the piston and rocker arm.' name='description'/>
8      <meta content='Todd Gagnon and Mark A. Boyd' name='authors'/>
9      <meta content='Xeena VRML importer' name='translator'/>
10     <meta content='8 June 1998' name='created'/>
11     <meta content='20 December 2002' name='imported'/>
12     <meta content='3 February 2007' name='modified'/>
13     <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/Color.x3d' name='identifier'/>
14     <meta content='KelpTank.x3d' name='reference'/>
15     <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/PumpHouse.x3d' name='reference'/>
16     <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
17     <meta content='Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html' name='generator'/>
18     <meta content='../license.html' name='license'/>
19   </head>
20   <Scene>
21     <WorldInfo info='DTG of last update: 081200Jun98 Added: Updated: cycleInterval to 5.5 Modeled by: Todd Gagnon and Mark A. Boyd' title='pumpHou
22     <ExternProtoDeclare name='WhereAmI' url='"../Chapter14-Prototypes/WhereAmI.x3d#WhereAmI" "http://X3dGraphics.com/examples/X3dForWebAuthors/Cha
23     <ProtoInstance name='WhereAmI'/>
24     <Background skyColor='1 1 1'/>
25     <Viewpoint description='Book View' orientation='0 -1 0 0.53' position='-2.28 0.29 4.06'/>
26     <Group>
27       <Shape>
28         <Appearance DEF='pumpHouse'>
29           <Material diffuseColor='0.82 0.78 0.74'/>
30         </Appearance>
31         <IndexedFaceSet
32             ccw='true'
33             colorIndex='0 0 0 0 -1 0 0 1 1 -1 1 1 1 1 -1 1 0 0 1 -1 0 0 1 1 -1 1 1 1 1 -1 0 0 1 1 -1 0 0 1 1 -1 1 1 1 1 -1 0 0 0 0 -1 0 0 0 0 -1 0 0
34             colorPerVertex='true'
35             coordIndex='0 1 5 4 -1 5 1 2 6 -1 6 2 3 7 -1 3 0 4 7 -1 1 12 13 2 -1 2 13 14 -1 12 15 16 13 -1 15 0 3 16 -1 16 3 17 -1 9 5 6 10 -1 8
36           <Coordinate DEF='CoordinateNodeExample' point='0.0 0.0 0.0 2.0 0.0 0.0 2.0 1.75 0.0 0.0 1.75 0.0 0.0 0.625 0.75 0.0 1.0 0.75 0.0 1.0 1.6 0.0
37           <Color DEF='ColorNodeExample' color='.82 .78 .74 .66 .37 .02'/>
38         </IndexedFaceSet>
39       </Shape>
40       <Transform scale='0.91 0.6 0.3' translation='0.8 -0.65 0.5'>
41         <Shape>
42           <Appearance>
43             <Material diffuseColor='0.749 0.694 0.651'/>
44           </Appearance>
45           <Cylinder bottom='false' top='false'/>
46         </Shape>
```

color, coordinate correspondences

Edit Color

containerField: color

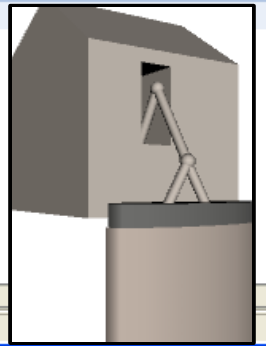
DEF: ColorNodeExample

USE:

color array

	r	g	b	color
0	0.82	0.78	0.74	
1	0.66	0.37	0.02	

OK Cancel Help



37:1 INS

Color	**Color node defines a set of RGB color values. Color is only used by ElevationGrid, IndexedFaceSet, IndexedLineSet, LineSet and PointSet.** **Hint: colors are often controlled by Material instead.**
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!
color	**[color: accessType inputOutput, type MFColor CDATA #IMPLIED]** color defines a set of RGB colors.
containerField	**[containerField: NMTOKEN "color"]** 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.

ColorRGBA	**ColorRGBA node defines a set of RGBA color values. ColorRGBA is only used by ElevationGrid, IndexedFaceSet, IndexedLineSet, LineSet and PointSet.** **Hint: colors are often controlled by Material instead.** **Hint: alpha channel may be ignored under Interchange profile.**
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!
color	**[color: accessType inputOutput, type MFColorRGBA CDATA #IMPLIED]** color defines a set of RGBA colors.
containerField	**[containerField: NMTOKEN "color"]** 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.

PointSet node

PointSet creates a series of simple unconnected points in 3D space

- Contains Coordinate node for *point* data
- Since points are separate, *coordIndex* unnecessary

Each point typically drawn as a single pixel

- Or consistently as multiple pixels
- Thus scaling and perspective are quite deceiving
- Rarely used due to perspective inconsistencies

Color can be set in one of two ways

- Uniformly via Material *emissiveColor* value
- Individually via contained Color/ColorRGBA node

PointSet.x3d - Editor

PointSet.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/s
  <head>
    <meta content='PointSet.x3d' name='title'/>
    <meta content='Way points for the animated shark Lucy traversing the tank.' name='description'/>
    <meta content='Tim McLean' name='creator'/>
    <meta content='Don Brutzman' name='translator'/>
    <meta content='June 1998' name='created'/>
    <meta content='20 December 2002, 11 March 2006' name='modified'/>
    <meta content='http://web.nps.navy.mil/~brutzman/kelp' name='reference'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/SharkLucyLocale.x3d' name='reference'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/PointSet.x3d' name='identifier'/>
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <ExternProtoDeclare name='WhereAmI' url='"../Chapter14-Prototypes/WhereAmI.x3d#WhereAmI" "http://X3dGraphics.com/examples/X3dForWebAuthors/Cha
    <ProtoInstance name='WhereAmI'/>
    <Background skyColor='0 0 0'/>
    <Viewpoint description='Book View' orientation='0.939 0.335 0.075 -0.57' position='-0.89 1.91 9.26'/>
    <Transform DEF='_0' translation='0.0 -8.0 -1.0'>
      <Inline url='"SharkLucy.wrl" "http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/SharkLucy.wrl" "SharkLucy.x3d" "http://X3dG
      <Group>
        <TimeSensor DEF='SHARK1_CLOCK' cycleInterval='220.0' loop='true'/>
        <PositionInterpolator DEF='SHARK1_POSITION' key='0.0 0.048 0.112 0.155 0.184 0.263 0.3 0.342 0.375 0.404 0.457 0.497 0.57 0.65 0.702 0.796
        <OrientationInterpolator DEF='SHARK1_ORIENTATION' key='0.0 0.048 0.112 0.155 0.184 0.263 0.3 0.342 0.375 0.404 0.457 0.497 0.57 0.65 0.702
      </Group>
    </Transform>
    <Transform translation='0 0 0'>
      <Shape>
        <Appearance>
          <Material emissiveColor='1 1 0'/>
        </Appearance>
        <PointSet>
          <Coordinate point='0.0 -7.0 -1.0 -1.75 -7.0 -0.5 -4.0 -7.0 0.5 -5.0 -6.5 1.5 -5.5 -6.25 0.75 -5.25 -5.5 -2.25 -4.25 -5.0 -3.25 -2.75 -4.
        </PointSet>
      </Shape>
    </Transform>
    <TimeSensor DEF='_4' loop='true'/>
    <Script DEF='sharkSwimmingInTankTrigger_5'>
      <field accessType='inputOnly' name='triggerIn' type='SFTime'/>
      <field accessType='outputOnly' name='startTime' type='SFTime'/>
      <field accessType='outputOnly' name='firstTime' type='SFBool'/>
      <![CDATA[ecmascript:
```

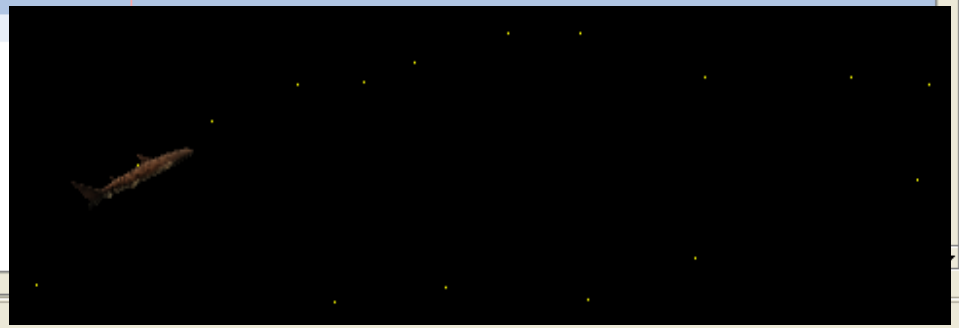
Edit PointSet

containerField: geometry

DEF

USE

OK Cancel Help



38:1 INS

PointSet	**PointSet is a node that contains a set of colored 3D points, represented by contained Color and Coordinate nodes. Color values or a Material emissiveColor is used to draw lines and points.** **Hint: use a different color (or emissiveColor) than the background color.** **Hint: insert a Shape node before adding geometry or Appearance. You can also substitute a type-matched ProtoInstance for content.**
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!
containerField	**[containerField: NMTOKEN "geometry"]** 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.

IndexedLineSet node

IndexedLineSet creates an array of line segments

- Contains Coordinate node for *point* data
- Can be discontinuous or share points repeatedly
- Each set of connected line segments is a *polyline*

Lines are not lit, use no texture-mapped images, and do not participate in collision detection

Color can be set in one of two ways

- Uniformly via Material *emissiveColor* value

Not *diffuseColor*!

- Individually via contained Color/ColorRGBA node; applied either by individual points, or by each segment, as determined by *colorPerVertex*

IndexedLineSet.x3d - Editor

IndexedLineSet.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLoca
  <head>
    <meta content='IndexedLineSet.x3d' name='title'/>
    <meta content='The path of the animated shark Lucy traversing the tank.' name='description'/>
    <meta content='Tim McLean' name='creator'/>
    <meta content='Don Brutzman' name='translator'/>
    <meta content='June 1998' name='created'/>
    <meta content="3 February 2008" name="modified"/>
    <meta content='http://web.nps.navy.mil/~brutzman/kelp' name='reference'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/SharkLucyLocale.x3d' name='refer
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/IndexedLineS
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <ExternProtoDeclare name='WhereAmI' url='"../Chapter14-Prototypes/WhereAmI.x3d#WhereAmI"
    <ProtoInstance name='WhereAmI'/>
    <Background skyColor='1 1 1'/>
    <Viewpoint description='Book View' orientation='0.939 0.335 0.075 -0.57' position='-0.89
    <Transform translation='0 0 0'>
      <Shape>
        <Appearance>
          <Material emissiveColor='0 0 1'/>
        </Appearance>
        <IndexedLineSet DEF='ILS' coordIndex='0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 0 -1'>
          <Coordinate point='0.0 -7.0 -1.0 -1.75 -7.0 -0.5 -4.0 -7.0 0.5 -5.0 -6.5 1.5 -5.5 -6.25 0.75 -5.25 -5.5 -2.25 -4.25 -5.0 -3.25 -2.75 -4.
        </IndexedLineSet>
      </Shape>
    </Transform>
    <Transform DEF='_0' translation='0.0 -8.0 -1.0'>
      <Inline url='"SharkLucy.wrl" "http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForest
      <Group>
        <TimeSensor DEF='SHARK1_CLOCK' cycleInterval='220.0' loop='true'/>
        <PositionInterpolator DEF='SHARK1_POSITION' key='0.0 0.048 0.112 0.155 0.184 0.263 0.3
        <OrientationInterpolator DEF='SHARK1_ORIENTATION' key='0.0 0.048 0.112 0.155 0.184 0.26
      </Group>
    </Transform>
    <TimeSensor DEF='_4' loop='true'/>
    <Script DEF='sharkSwimmingInTankTrigger_5'>
      <field accessType='inputOnly' name='triggerIn' type='SFTime'/>
      <field accessType='outputOnly' name='startTime' type='SFTime'/>
      <field accessType='outputOnly' name='firstTime' type='SFBool'/>
      <![CDATA[ecmascript:
```

27:24 INS

Edit Coordinate

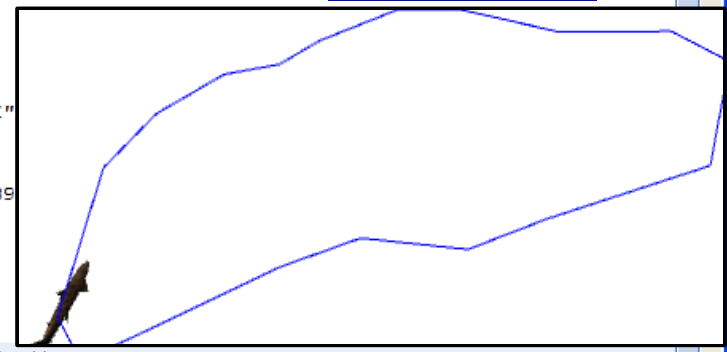
DEF TurnPoints

USE

point array

	x	y	z
0	0	-7	-1
1	-1.75	-7	-0.5
2	-4	-7	0.5
3	-5	-6.5	1.5
4	-5.5	-6.25	0.75
5	-5.25	-5.5	-2.25
6	-4.25	-5	-3.25
7	-2.75	-4.5	-3.75
8	-1.5	-4.5	-4
9	-0.5	-4.25	-4.5
10	1.5	-3.75	-4.75
11	3	-3.75	-4.5
12	5.75	-4.5	-4.5
13	8.75	-4.5	-4
14	9.25	-4.5	-2.25
15	7.5	-5.5	0
16	4	-6.5	-0.25
17	2.25	-7	-0.25

OK Cancel Help



Edit IndexedLineSet

DEF ILS

USE ILS

containerField

geometry

colorIndex

colorPerVertex

coordIndex 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 0 -1

Closed loop

OK Cancel Help

IndexedLineSet	**IndexedLineSet is a geometry node that can contain a Color node and a Coordinate node. Color values or a Material emissiveColor is used to draw lines and points. Lines are not lit, are not texture-mapped, and do not participate in collision detection.** **Hint: use a different color (or emissiveColor) than the background color.** **Hint: if rendering Coordinate points originally defined for an IndexedFaceSet, index values may need to repeat each initial vertex to close each polygon outline. Step-wise colors or linear interpolation of colors can be used as a good scientific visualization technique to map arbitrary function values to a color map.** **Hint: insert a Shape node before adding geometry or Appearance. You can also substitute a type-matched ProtoInstance for content.**
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!
coordIndex	**[coordIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** coordIndex indices provide order in which coordinates are applied. Order starts at index 0, commas are optional between sets, use -1 to separate indices for each polyline. Hint: if rendering Coordinate points originally defined for an IndexedFaceSet, index values may need to repeat initial each initial vertex to close the polygons.
colorPerVertex	**[colorPerVertex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether Color node is applied per vertex (true) or per polyline (false).
colorIndex	**[colorIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** colorIndex indices provide order in which colors are applied. Hint: if rendering Coordinate points originally defined for an IndexedFaceSet, index values may need to repeat initial each initial vertex to close the polygons.
set_coordIndex	**[set_coordIndex: accessType inputOnly, type MFInt32 CDATA #FIXED ""]** coordIndex indices provide order in which coordinates are applied. Order starts at index 0, commas are optional between sets. Use -1 to separate indices for each polygon.
set_colorIndex	**[set_colorIndex: accessType initializeOnly, type MFInt32 CDATA #FIXED ""]** colorIndex indices provide order in which colors are applied.
containerField	**[containerField: NMTOKEN "geometry"]** 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.

LineSet node

Similar to IndexedLineSet

- Contain 0 or 1 Coordinate/CoordinateDouble
- Material *emissiveColor* or Color/ColorRGBA

Rather than using coordIndex and colorIndex, LineSet has *vertexCount* field

- *vertexCount* MFInt32 array of integers defines number of sequential points used in each polyline
- No -1 sentinel values needed
- Color and Coordinate values used in defined order
- Somewhat more compact than using indices

LineSet.x3d - Editor

LineSet.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchem
  <head>
    <meta content='LineSet.x3d' name='title'/>
    <meta content='The way points and path of the animated shark Lucy traversing the tank.' name='description'/>
    <meta content='Tim McLean' name='creator'/>
    <meta content='Don Brutzman' name='translator'/>
    <meta content='June 1998' name='created'/>
    <meta content='20 December 2002, 11 March 2006' name='modified'/>
    <meta content='http://web.nps.navy.mil/~brutzman/kelp' name='reference'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/SharkLucyLocale.x3d' name='
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/LineSet
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <ExternProtoDeclare name='WhereAmI' url='"../Chapter14-Prototypes/WhereAmI.x3d#WhereAmI" "http://X3dGraphics.
    <ProtoInstance name='WhereAmI'/>
    <Background skyColor='1 1 1'/>
    <Viewpoint description='Book View' orientation='0.939 0.335 0.075 -0.57' position='-0.89 1.91 9.26'/>
    <Transform DEF='_0' translation='0.0 -8.0 -1.0'>
      <Inline url='"SharkLucy.wrl" "http://X3dGraphics.com/examples/
      <Group>
        <TimeSensor DEF='SHARK1_CLOCK' cycleInterval='220.0' loop='t
        <PositionInterpolator DEF='SHARK1_POSITION' key='0.0 0.048 0
        <OrientationInterpolator DEF='SHARK1_ORIENTATION' key='0.0 0
      </Group>
    </Transform>
    <Transform translation='0 0 0'>
      <Shape>
        <Appearance>
          <Material emissiveColor='0 0 1'/>
        </Appearance>
        <LineSet vertexCount='19'>
          <Coordinate point='0.0 -7.0 -1.0 -1.75 -7.0 -0.5 -4.0 -7.0 0.5 -5.0 -6.5 1.5 -5.5 -6.25 0.75 -5.25 -5.5 -2.25 -4.25 -5.0 -3.25 -2.75 -4.5
        </LineSet>
      </Shape>
      <Shape>
        <Appearance>
          <Material emissiveColor='1 0 0'/>
        </Appearance>
        <PointSet>
          <Coordinate point='0.0 -7.0 -1.0 -1.75 -7.0 -0.5 -4.0 -7.0 0.5 -5.0 -6.5 1.5 -5.5 -6.25 0.75 -5.25 -5.5 -2.25 -4.25 -5.0 -3.25 -2.75 -4.5
        </PointSet>
      </Shape>
```

Edit Coordinate

DEF TurnPoints

USE

point array

	x	y	z
0	0	-7	-1
1	-1.75	-7	-0.5
2	-4	-7	0.5
3	-5	-6.5	1.5
4	-5.5	-6.25	0.75
5	-5.25	-5.5	-2.25
6	-4.25	-5	-3.25
7	-2.75	-4.5	-3.75
8	-1.5	-4.5	-4
9	-0.5	-4.25	-4.5
10	1.5	-3.75	-4.75
11	3	-3.75	-4.5
12	5.75	-4.5	-4.5
13	8.75	-4.5	-4
14	9.25	-4.5	-2.25
15	7.5	-5.5	0
16	4	-6.5	-0.25
17	2.25	-7	-0.25
18	0	-7	-1

OK Cancel Help

Edit LineSet

DEF

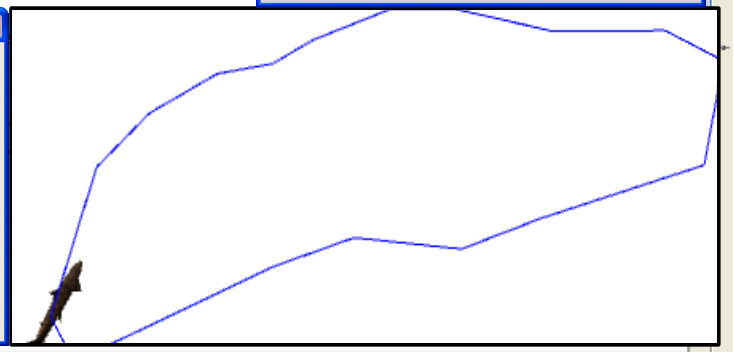
USE

containerField

geometry

vertexCount 19

OK Cancel Help



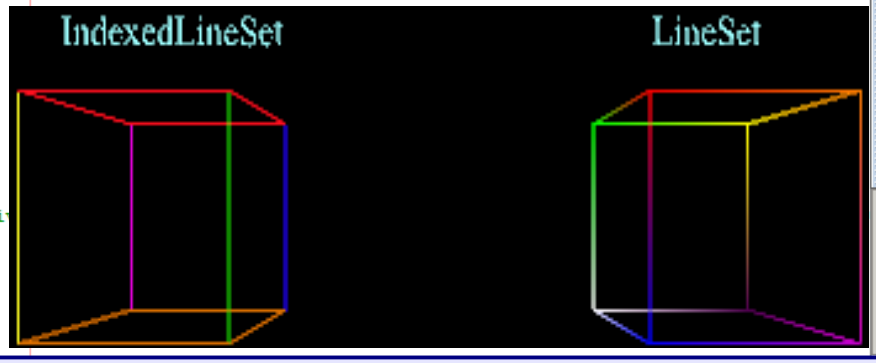
35:17 INS

LineSetComparison.x3d - Editor

LineSetComparison.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile='Immersive' version='3.0' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/speci
    <head>
        <meta content='LineSetComparison.x3d' name='title'/>
        <meta content='Don Brutzman' name='creator'/>
        <meta content='2 September 2008' name='created'/>
        <meta content='2 September 2008' name='modified'/>
        <meta content='A simple wireframe cube, comparing IndexedLineSet (per-vertex coloring) with LineSet (per-segment coloring).' name='description
        <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/LineSetComparison.x3d' name='identifier'
        <meta content='http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook/Chapter13-PointsLinesFaces/Figure13.11IndexedLineSetBoxWireframe.x3
        <meta content='X3D-Edit 3.2, https://savage.nps.edu/X3D-Edit' name='generator'/>
        <meta content='../../license.html' name='license'/>
    </head>
    <Scene>
        <Viewpoint description='8m away - note linewidth is unchanged and perspective is ambiguous' position='0 0 8'/>
        <Viewpoint description='15m away - note linewidth is unchanged and perspective is ambiguous' position='0 0 15'/>
        <Transform translation="-3 0 0">
            <Shape>
                5       5       2   2   2   2
                <IndexedLineSet colorPerVertex="false" coordIndex="0 1 2 3 0 -1 4 5 6 7 4 -1 0 4 -1 1 5 -1 2 6 -1 3 7">
                    <Coordinate point="-1 1 1 1 1 1 1 1 -1 -1 1 -1 -1 -1 1 1 -1 1 1 -1 -1 -1 -1 -1"/>
                    <Color color="1 0.058824 0.117647, 0.878431 0.447059 0, 1 0.992157 0.141176, 0.101961 0.721569 0, 0.12549 0 0.901961, 0.878431 0 0
                </IndexedLineSet>
            </Shape>
        </Transform>
        <Transform translation="3 0 0">
            <Shape>
                5       5       2   2   2   2
                <!-- Coordinate sequence (matches coordIndex actually) 0 1 2 3 0, 4 5 6 7 4, 0 4, 1 5, 2 6, 3 7 -->
                <LineSet vertexCount='5 5 2 2 2 2'>
                    <Coordinate point='-1 1 1, 1 1 1, 1 1 -1, -1 1 -1, -1 1 1, -1 -1 1, 1 -1 1, 1 -1 -1, -1 -1 -1, -1 -1 1, -1 1 1, -1 -1 1, 1 1 1, 1
                    <Color color='1 0 0, 1 0.5 0, 1 1 0, 0 1 0, 1 0 0, 0 0 1, 0.8 0 0.8, 0.3 0 0.3, 1 1 1, 0 0 1, 1 0 0, 0 0 1, 1 0.5 0, 0.8 0 0.8, 1
                </LineSet>
            </Shape>
        </Transform>
        <Transform translation="-3 2 0">
            <Shape>
                <Text string='IndexedLineSet'>
                    <FontStyle size="0.5" justify='"MIDDLE" "MIDDLE"'/>
                </Text>
                <Appearance DEF="TextAppearance">
                    <Material ambientIntensity='0' diffuseColor='0 0 0' emissi
                </Appearance>
            </Shape>
        </Transform>
        <Transform translation="3 2 0">
            <Shape>
                <Text string='LineSet'>
```

Note pattern in order, count



colorPerVertex='false' per-vertex Color

5:45 INS

LineSet	**LineSet is a geometry node that can contain a Color node and a Coordinate node. Color values or a Material emissiveColor is used to draw lines and points. Lines are not lit, are not texture-mapped, and do not participate in collision detection.** **Hint: use a different color (or emissiveColor) than the background color. Linear interpolation of colors can be used as a good scientific visualization technique to map arbitrary function values to a color map.** **Hint: insert a Shape node before adding geometry or Appearance. You can also substitute a type-matched ProtoInstance for content.**
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!
vertexCount	**[vertexCount: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** [2,infinity) vertexCount describes how many vertices are used in each polyline from Coordinate field. Coordinates are assigned to each line by taking vertexCount[n] vertices from Coordinate field.
containerField	**[containerField: NMTOKEN "geometry"]** 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.

IndexedFaceSet node 1

IndexedFaceSet creates a set of polygons (faces)

- Contains Coordinate node for *point* data
- Can be discontinuous or share points repeatedly
- You can essentially create any geometry with IFS

Color can be set in one of two ways

- Uniformly via sibling Material fields
- Individually via contained Color/ColorRGBA node; applied either by individual points, or by each polygon, as determined by *colorPerVertex*

IndexedFaceSet node 2

Many fields and features apply

- *ccw*, *convex*, *solid*, *creaseAngle* as before
- *colorPerVertex*, *normalPerVertex* as before
- *coordIndex, colorIndex, normalIndex* as before
- *texCoordIndex* applies texture coordinates to map texture images to individual geometry points

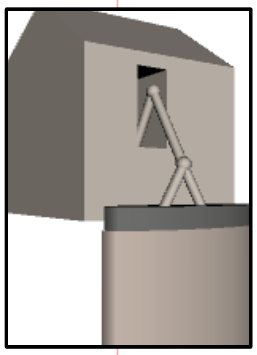
Contained nodes (0 or 1 of each)

- Coordinate/CoordinateDouble (essential, required)
- Color/ColorRGBA
- Normal, TextureCoordinate

IndexedFaceSet.x3d - Editor

IndexedFaceSet.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/spe
  <head>
    <meta content='IndexedFaceSet.x3d' name='title'/>
    <meta content='Positive-displacement cylinder pump house built using IndexedFaceSet nodes.' name='description'/>
    <meta content='Todd Gagnon and Mark A. Boyd' name='authors'/>
    <meta content='Xeena VRML importer' name='translator'/>
    <meta content='8 June 1998' name='created'/>
    <meta content='20 December 2002' name='imported'/>
    <meta content='7 March 2006' name='modified'/>
    <meta content='KelpTank.x3d' name='reference'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/PumpHouse.x3d' name='reference'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/IndexedFaceSet.x3d' name='identifier'/>
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='Vrml97ToX3dNist, http://ovrt.nist.gov/v2_x3d.html' name='generator'/>
    <meta content='../license.html' name='license'/>
    <meta content='This is not really static because the PositionInterpolators contain an initial offest of the piston and rocker arm.' name='comment
  </head>
  <Scene>
    <ExternProtoDeclare name='WhereAmI' url='"../Chapter14-Prototypes/WhereAmI.x3d#WhereAmI" "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapte
    <ProtoInstance name='WhereAmI'/>
    <Background skyColor='1 1 1'/>
    <Viewpoint description='Book View' orientation='0 -1 0 0.53' position='-2.28 0.29 4.06'/>
    <Group>
      <Shape>
        <Appearance DEF='pumpHouse'>
          <Material diffuseColor='0.82 0.78 0.74'/>
        </Appearance>
        <IndexedFaceSet DEF='IFS' coordIndex='0 1 5 4 -1 5 1 2 6 -1 6 2 3 7 -1 3 0 4 7 -1 1 12
          <Coordinate point='0.0 0.0 0.0 2.0 0.0 0.0 2.0 1.75 0.0 0.0 1.75 0.0 0.625 0.75 0.0 1
        </IndexedFaceSet>
      </Shape>
      <Transform scale='0.91 0.6 0.3' translation='0.8 -0.65 0.5'>
        <Shape>
          <Appearance>
            <Material diffuseColor='0.749 0.694 0.651'/>
          </Appearance>
          <Cylinder bottom='false' top='false'/>
        </Shape>
      </Transform>
    </Group>
    <Group>
      <Transform scale='0.5 0.5 0.5' translation='1.0 1.1 -1.5'>
        <Transform DEF='PISTON'>
          <Transform scale='1.8 1.2 0.6' translation='0.0 -0.2 0.0'>
```



Edit IndexedFaceSet

DEF IFS
USE IFS
containerField
geometry

ccw
convex
solid
creaseAngle 0
colorPerVertex
normalPerVertex
colorIndex
coordIndex 0 1 5 4 -1 5 1 2 6 -1 6 2 3 7 -1 3 0 4 7 -1 1 12 13 2 -1 2 13 14 -1 12 15 16 13 -1 15 0 3 16 -1 16 3 17 -1 9 5 6 10 -1 8 9 10 11 -1 4 8 11
normalIndex
texCoordIndex

OK Cancel Help

30:24 INS

IndexedFaceSet	**IndexedFaceSet is a geometry node that can contain a Color, Coordinate, Normal and TextureCoordinate node.** **Hint: insert a Shape node before adding geometry or Appearance. You can also substitute a type-matched ProtoInstance for content.**
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!
coordIndex	**[coordIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** coordIndex indices provide order in which coordinates are applied. Order starts at index 0, commas are optional between sets. Use -1 to separate indices for each polygon.
ccw	**[ccw: accessType initializeOnly, type SFBool (true|false) "true"]** ccw = counterclockwise: ordering of vertex coordinates orientation. **Hint:** ccw false can reverse solid (backface culling) and normal-vector orientation.
convex	**[convex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether all polygons in a shape are convex (true), or possibly concave (false) A convex polygon is planar, does not intersect itself, and has all interior angles < 180 degrees. **Interchange profile hint:** only convex=true IndexedFaceSets are supported. **Warning:** concave geometry may be invisible default convex=true.
solid	**[solid: accessType initializeOnly, type SFBool (true|false) "true"]** Setting solid true means draw only one side of polygons (backface culling on), setting solid false means draw both sides of polygons (backface culling off). **Warning:** default value true can completely hide geometry if viewed from wrong side!
creaseAngle	**[creaseAngle: accessType initializeOnly, type SFFloat CDATA "0"]** [0..infinity) creaseAngle defines angle (in radians) for determining whether adjacent polygons are drawn with sharp edges or smooth shading. If angle between normals of two adjacent polygons is less than creaseAngle, smooth shading is rendered across the shared line segment. **Interchange profile hint:** only 0 and Î radians supported. **Hint:** creaseAngle=0 means render all edges sharply, creaseAngle=3.14 means render all edges smoothly.
colorPerVertex	**[colorPerVertex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether Color node is applied per vertex (true) or per polygon (false).
colorIndex	**[colorIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** colorIndex indices provide order in which colors are applied.
normalPerVertex	**[normalPerVertex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether Normal node is applied per vertex (true) or per polygon (false).
normalIndex	**[normalIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** **Interchange profile hint:** this field may be ignored.

texCoordIndex	**[texCoordIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** List of texture-coordinate indices mapping attached texture to corresponding coordinates. **Hint:** use a tool!
set_coordIndex	**[set_coordIndex: accessType inputOnly, type MFInt32 CDATA #FIXED ""]** coordIndex indices provide order in which coordinates are applied. Order starts at index 0, commas are optional between sets. Use -1 to separate indices for each polygon.
set_colorIndex	**[set_colorIndex: accessType initializeOnly, type MFInt32 CDATA #FIXED ""]** colorIndex indices provide order in which colors are applied.
set_normalIndex	**[set_normalIndex: accessType inputOnly, type MFInt32 CDATA #FIXED ""]** **Interchange profile hint:** this field may be ignored.
set_texCoordIndex	**[set_texCoordIndex: accessType inputOnly, type MFInt32 CDATA #FIXED ""]** List of texture-coordinate indices mapping attached texture to corresponding coordinates. **Hint:** use a tool!
containerField	**[containerField: NMTOKEN "geometry"]** 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.

ElevationGrid node

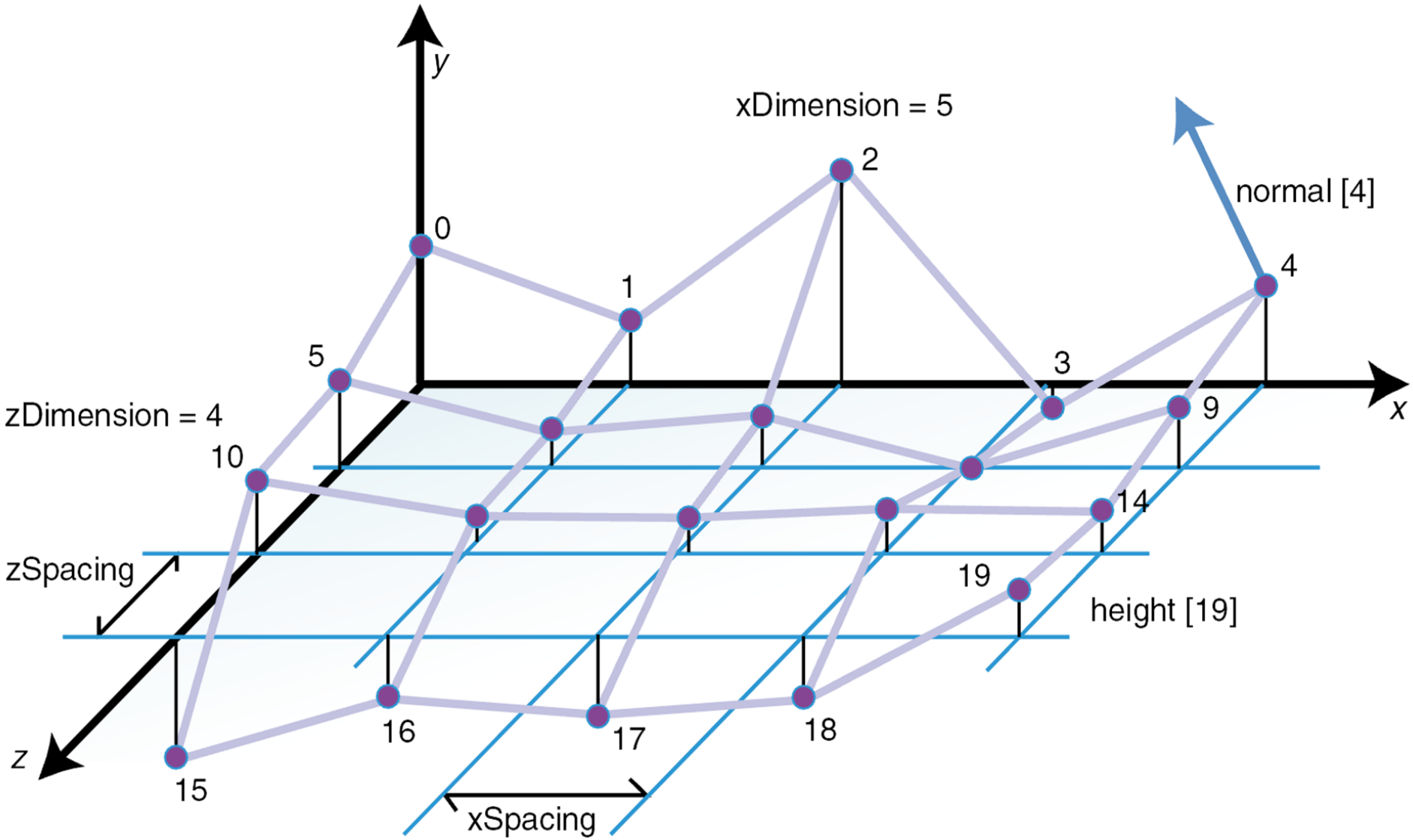
ElevationGrid takes a rectangular array of floats and converts *height* array into post values above (or below) baseline y=0 ground plane

- *xDimension, zDimension* are row, column sizes
- *xSpacing*, *zSpacing* are lengths in meters
- *height* MFFloat array (size *xDimension · zDimension*)
- *ccw*, *solid* as before
- *colorPerVertex*, *normalPerVertex* as before

Contained nodes (0 or 1 of each)

- Color/ColorRGBA, Normal, TextureCoordinate

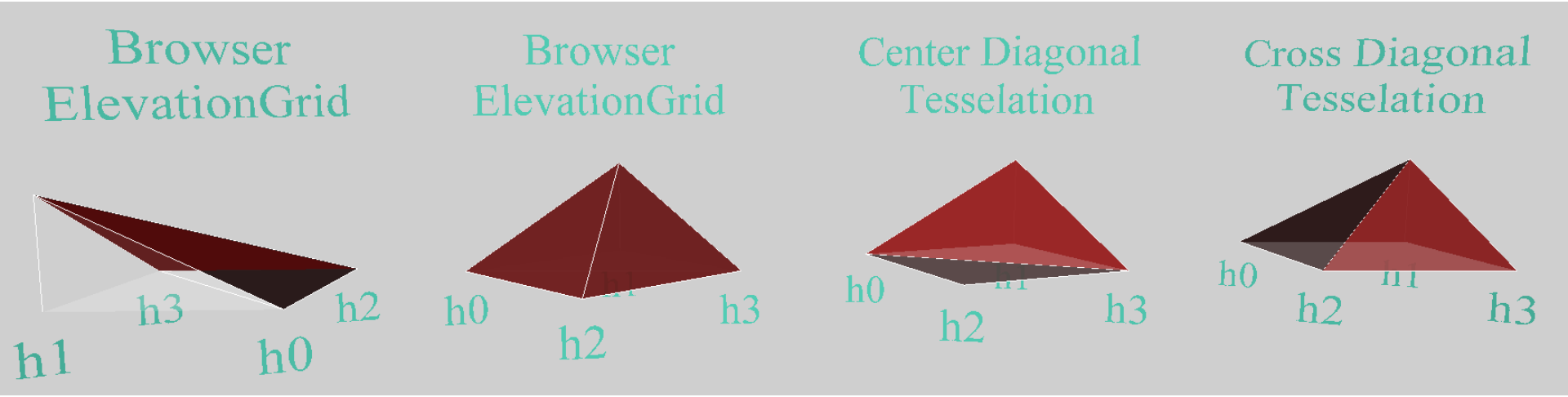
ElevationGrid indexing of *height* array



ElevationGrid inconsistencies due to noncoplanar quadrilaterals!

Alternate forms of tessellation are possible for nonplanar ElevationGrid quadrilaterals

- Almost all ElevationGrid quads are nonplanar, otherwise the geometry is flat
- Leftmost two figures show different views of grid
- Rightmost two figures show different tessellations
- Can avoid problem by using larger, fine-scaled grids



index counting checks

- *colorIndex* count must equal (**point** count - 1) when *colorPerVertex*='true', which is default

- *colorIndex* count must equal (**polygon** count - 1) when *colorPerVertex*='false' (i.e. color per polygon)

- **point** count = (xDimension * zDimension)

- **polygon** count = (xDimension-1) * (zDimension-1)

ElevationGrid node

Simple authoring trick

- Use a spreadsheet or some other simple tool to create a table of values, then cut/paste the values into the ElevationGrid *height* field
- Don't forget to also specify dimensions and spacing

Interesting authoring trick

- ElevationGrid does not have to lay flat on the horizontal plane, you can rotate it to another angle
- Example: stone canyon walls inside Kelp Forest exhibit

ElevationGrid.x3d [New] - Editor

ElevationGrid.x3d [New]

```
<?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='
  <head>
    <meta content='ElevationGrid.x3d' name='title'/>
    <meta content='Simple ElevationGrid example.' name='description'/>
    <meta content='Don Brutzman' name='creator'/>
    <meta content='Naval Postgraduate School' name='organization'/>
    <meta content='8 May 2008' name='created'/>
    <meta content='8 May 2008' name='modified'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ElevationGrid.x3d'
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <Background skyColor='1 1 1'/>
    <Viewpoint description='Overhead View' orientation='1 0 0 -1.57' position='0 8 0'/>
    <Viewpoint DEF='DefaultView' description='ElevationGrid example' position='0 0.2 6'/>
    <Transform translation='-3 0 -3'>
      <Shape DEF='ExampleElevationGridShape'>
        <ElevationGrid height='0 0 0 0 0 0 1 1 1 1 0 0 1 2 2 1 0 0 1 2 2 1 0 0 1 1 1 1 0 0 0 0 0 0 0' solid='false'
        xDimension='6' zDimension='6'/>
        <Appearance DEF='DiffuseRedAppearance'>
          <Material diffuseColor='0.9 0.1 0.1'/>
        </Appearance>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```

Edit ElevationGrid

containerField: geometry

DEF

USE

ccw ☑ solid ☐ creaseAngle 0

colorPerVertex ☑ xSpacing 1

normalPerVertex ☑ zSpacing 1

Height

6 columns (x dimension) Add column Remove column

1	2	3	4	5	6
0	0	0	0	0	0
0	1	1	1	1	0
0	1	2	2	1	0
0	1	2	2	1	0
0	1	1	1	1	0
0	0	0	0	0	0

6 rows (z dimension) Add row Remove row

OK Cancel Help

23:1 INS

ElevationGrid	**ElevationGrid is a geometry node. ElevationGrid is a rectangular grid of varying height above a flat surface. ElevationGrid can contain Color, Normal and TextureCoordinate nodes.** **Hint: insert a Shape node before adding geometry or Appearance. You can also substitute a type-matched ProtoInstance for content.**
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!
xDimension	**[xDimension: accessType initializeOnly, type SFInt32 CDATA "0"]** Number of grid-array elements along X direction.
zDimension	**[zDimension: accessType initializeOnly, type SFInt32 CDATA "0"]** Number of grid-array elements along Z direction.
xSpacing	**[xSpacing: accessType initializeOnly, type SFFloat CDATA "1.0"]** Meters distance between grid-array vertices along X direction. Hint: total horizontal x-axis distance equals (xDimension-1) * xSpacing.
zSpacing	**[zSpacing: accessType initializeOnly, type SFFloat CDATA "1.0"]** Meters distance between grid-array vertices along Z direction. Hint: total vertical z-axis distance equals (zDimension-1) * zSpacing.
height	**[height: accessType initializeOnly, type MFFloat CDATA #IMPLIED]** Grid array of height vertices along upward Y direction, with xDimension rows and zDimension columns.
set_height	**[set_height: accessType inputOnly, type MFFloat CDATA #FIXED ""]** Grid array of height vertices along upward Y direction, with xDimension rows and zDimension columns.
ccw	**[ccw: accessType initializeOnly, type SFBool (true|false) "true"]** ccw = counterclockwise: ordering of vertex coordinates orientation. Hint: ccw false can reverse solid (backface culling) and normal-vector orientation.
creaseAngle	**[creaseAngle: accessType initializeOnly, type SFFloat CDATA "0"]** [0..infinity) creaseAngle defines angle (in radians) for determining whether adjacent polygons are drawn with sharp edges or smooth shading. If angle between normals of two adjacent polygons is less than creaseAngle, smooth shading is rendered across the shared line segment. Hint: creaseAngle=0 means render all edges sharply, creaseAngle=3.14 means render all edges smoothly.
solid	**[solid: accessType initializeOnly, type SFBool (true|false) "true"]** Setting solid true means draw only one side of polygons (backface culling on), setting solid false means draw both sides of polygons (backface culling off). Warning: default value true can completely hide geometry if viewed from wrong side!
colorPerVertex	**[colorPerVertex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether Color node is applied per vertex (true) or per quadrilateral (false).
normalPerVertex	**[normalPerVertex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether Normal node is applied per vertex (true) or per quadrilateral (false).
containerField	**[containerField: NMTOKEN "geometry"]** 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.

Extrusion node 1

Extrusion begins with a planar *crossSection* outline, then stretches (extrudes) it along a *spine* polyline

- *crossSection* is MFVec2f array of 2-tuple floating-point 2D coordinate pairs creating an outline
- *spine* is MFVec3f array of 3-tuple floating-point 3D coordinates creating a polyline

Extrusion is a bit tricky to master, but provides a great way to create sophisticated geometry with little effort

Play-doh Fun Factory!

Here is an example real-world Extrusion

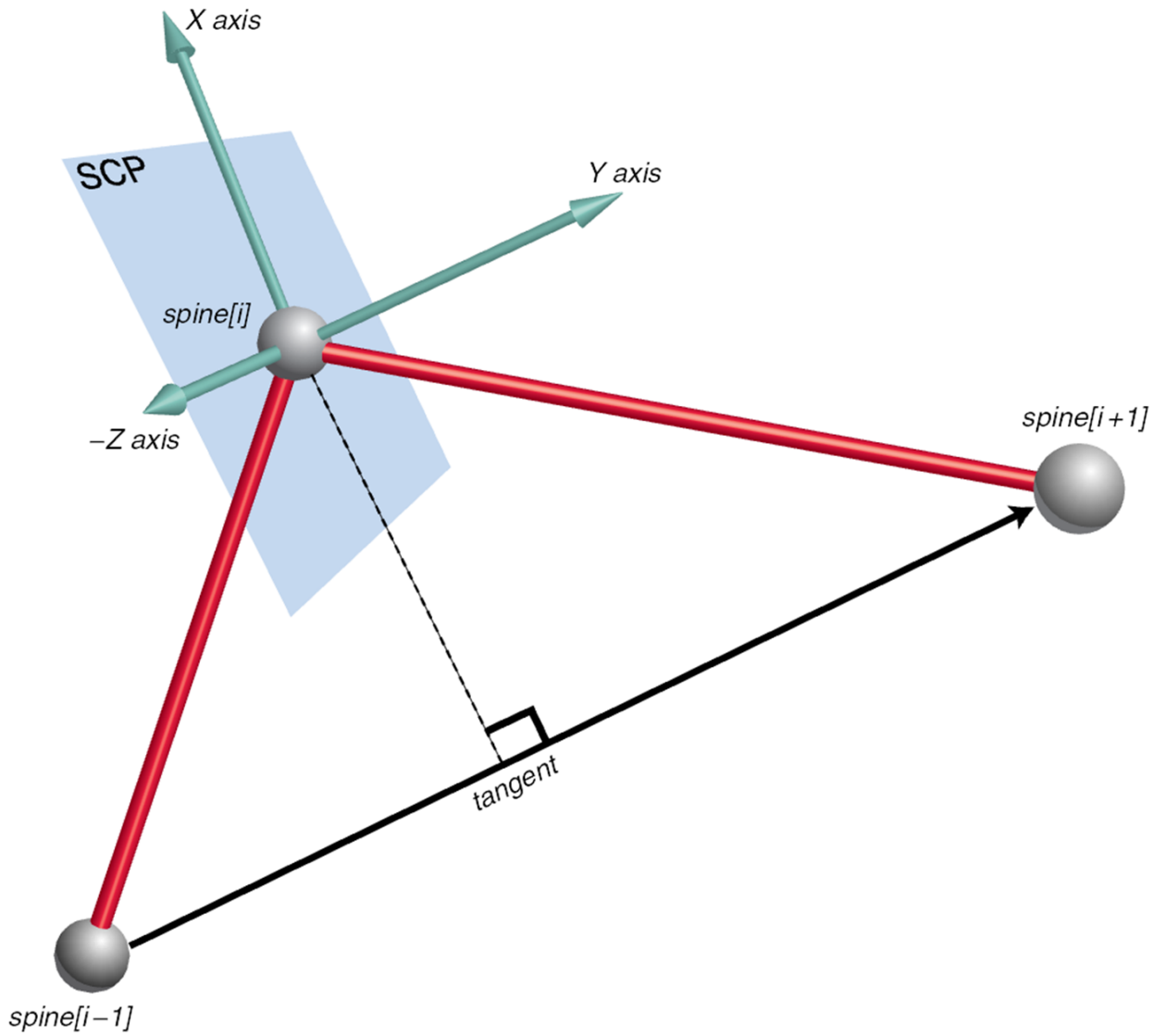
Extrusion node 2

Spine-aligned cross-section plane (SCP) refers to each individual copy of the cross section, each of which appear about each *spine* point

- Extrusion outer hull simply connects corresponding points on these cross-section outlines
- If the outline of the Extrusion is degenerate or ill defined, then the polygons making up Extrusion outline are similarly confounded

Drawing simple outlines of *crossSection* on graph paper is great way to keep things straight

Extrusion spine-aligned cross-section plane (SCP)



Extrusion node 2

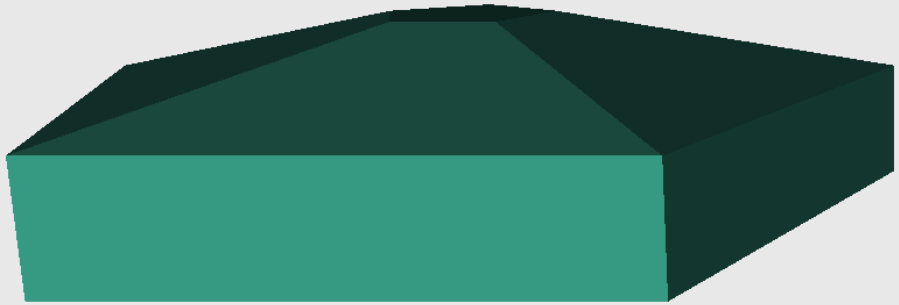
Further modifications include *scale*, *orientation* to modify each cross-section about SCP center

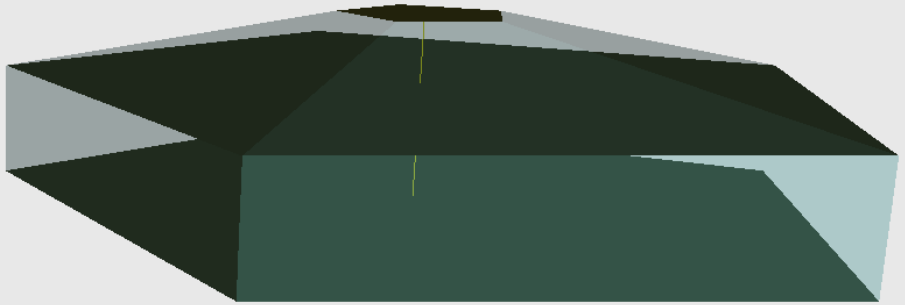
- *scale* is MFVec2f array of 2-tuple floating-point pairs to scale the local spine-aligned crossSection plane
- *orientation* is MFRotation array to rotate
- Single value affects all simultaneously, array affects each repeated cross-section individually

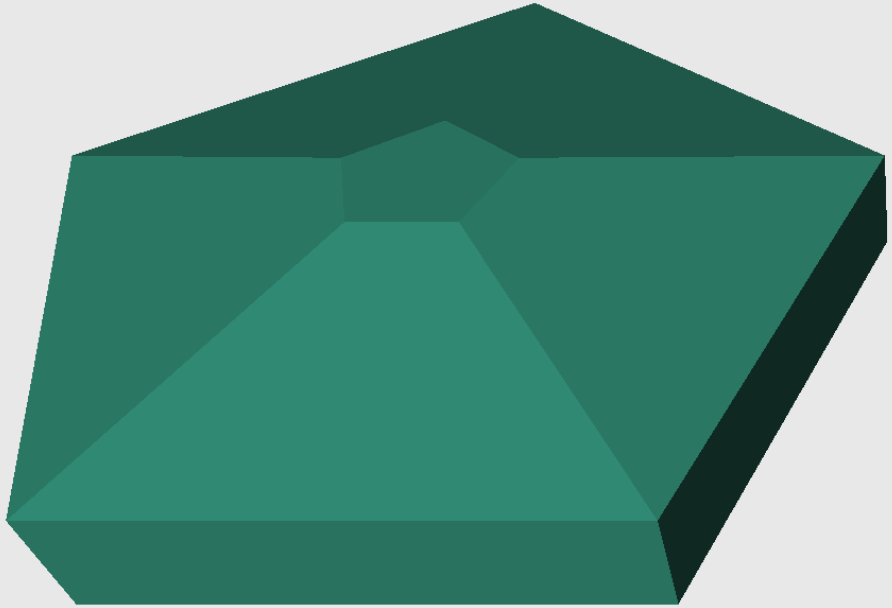
Other fields are common

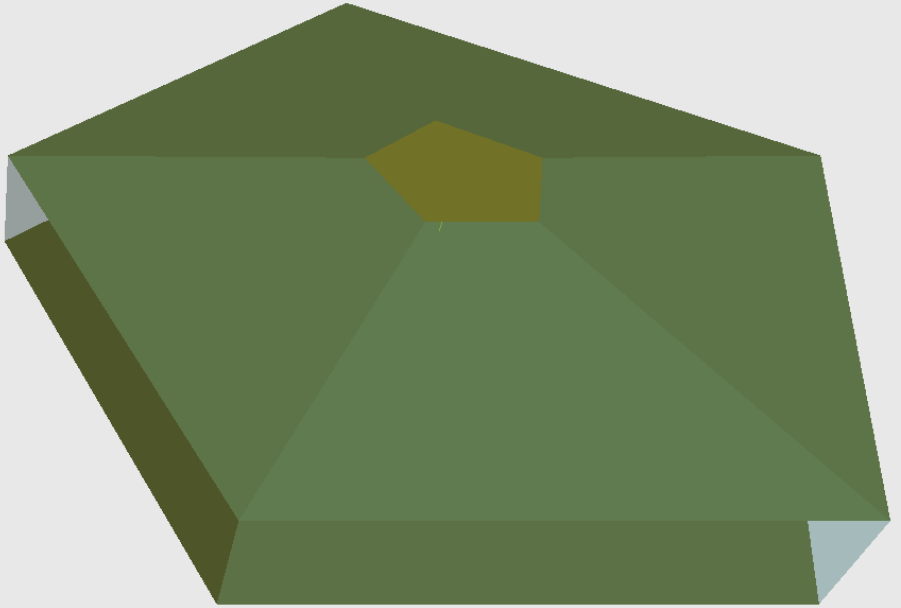
- *ccw*, *convex*, *solid*, *creaseAngle* as before
- *beginCap, endCap* are SFBool values to close ends

Extrusion cross-section example: pentagon





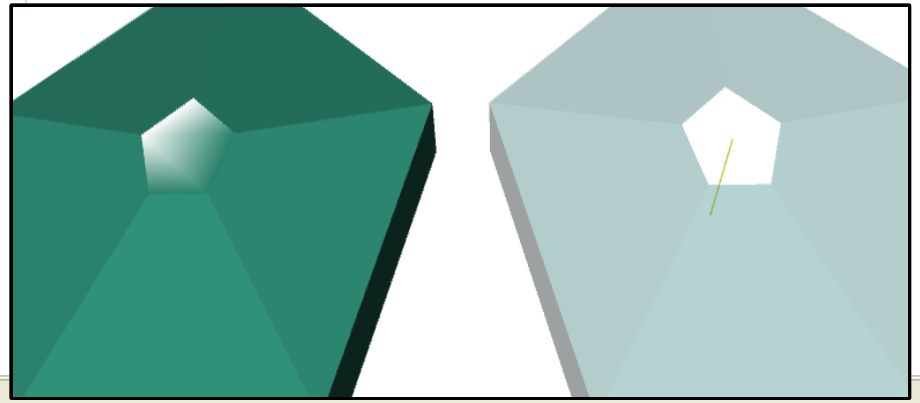




ExtrusionPentagon.x3d - Editor

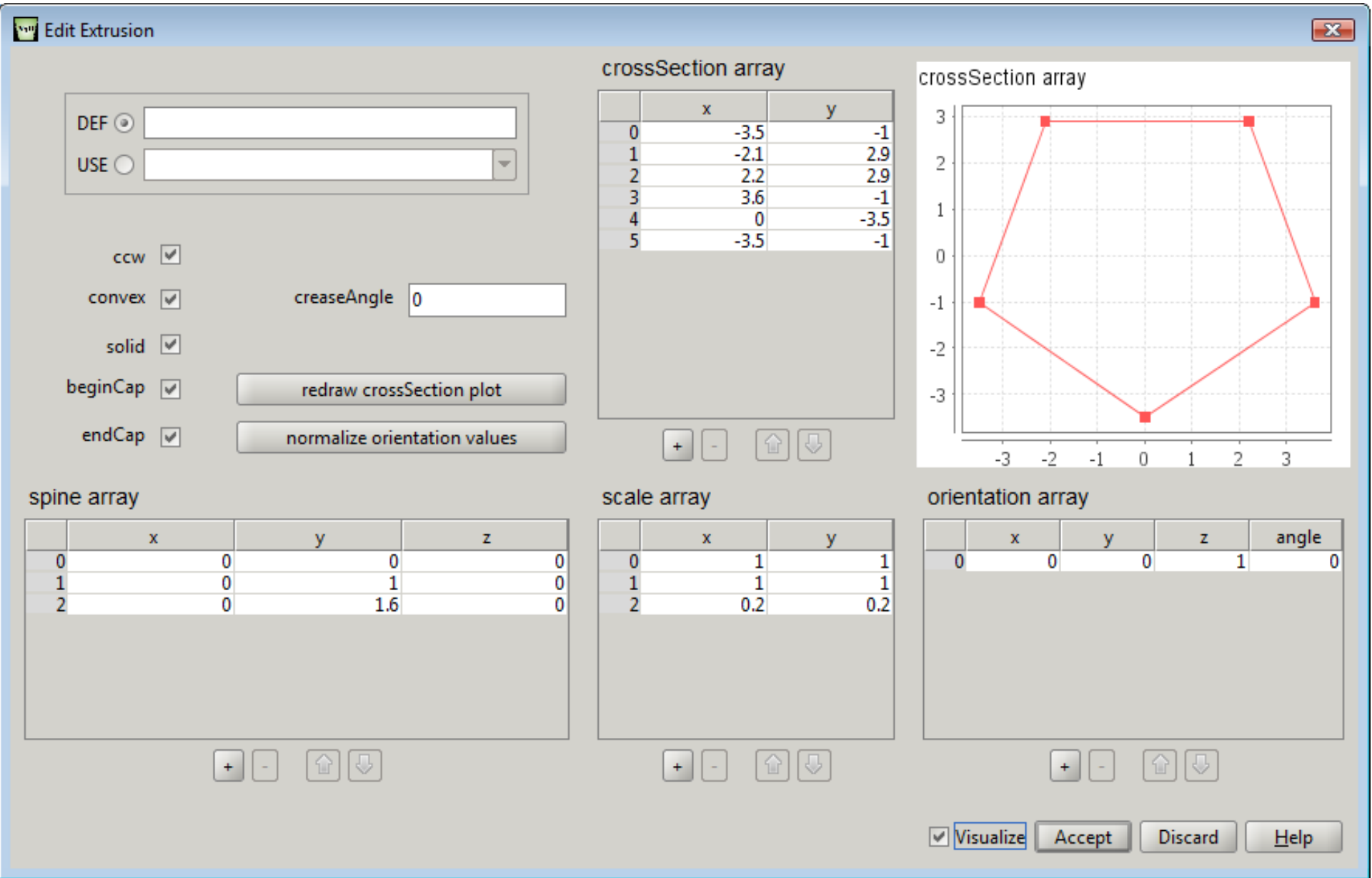
ExtrusionPentagon.x3d

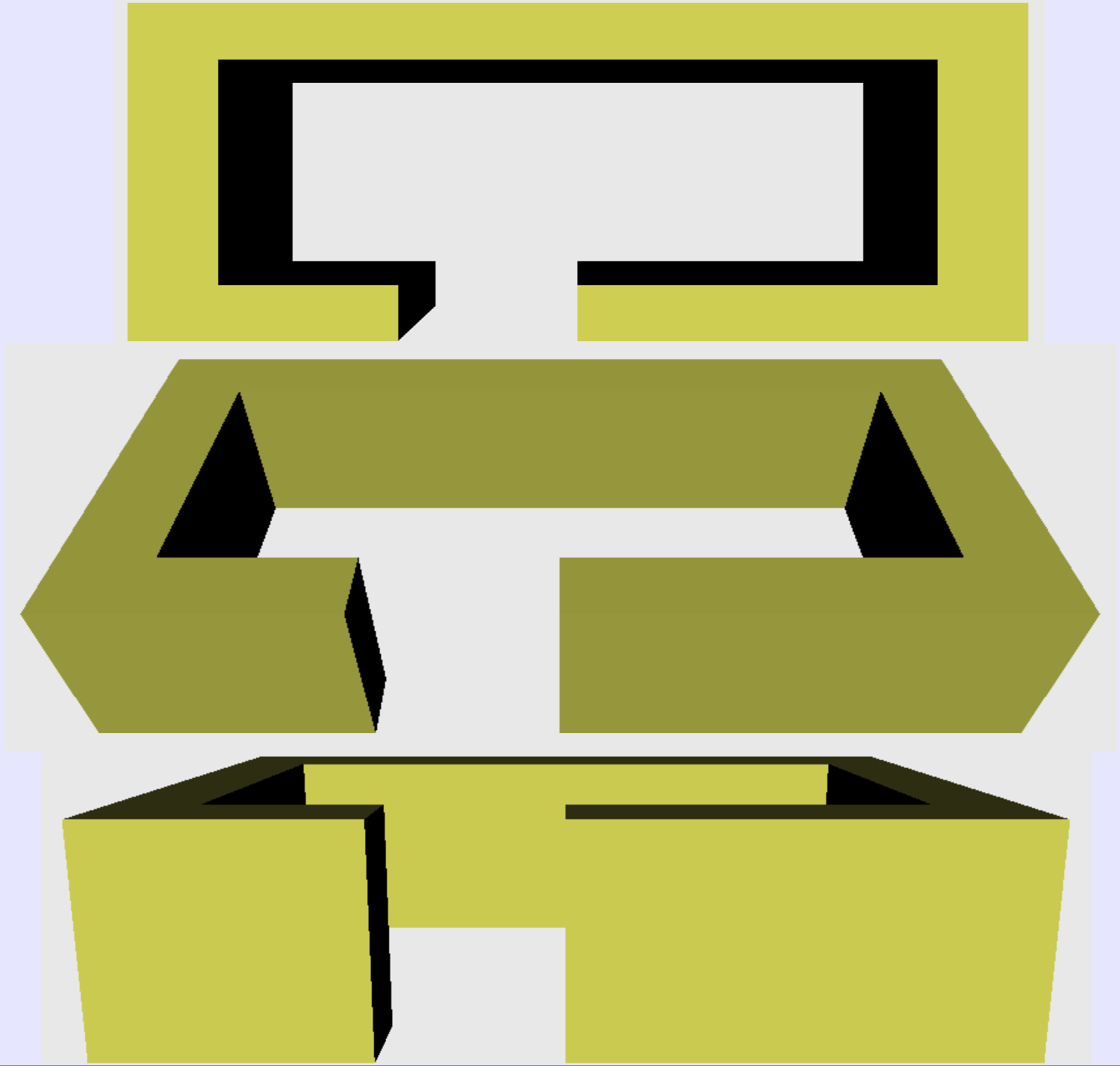
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.1.xsd'>
  <head>
    <meta content='ExtrusionPentagon.x3d' name='title'/>
    <meta content='Simple regular pentagon extruded vertically.' name='description'/>
    <meta content='Don Brutzman' name='creator'/>
    <meta content='Naval Postgraduate School' name='organization'/>
    <meta content='3 September 2005' name='created'/>
    <meta content='3 September 2005' name='modified'/>
    <meta content='Extrusion pentagon' name='subject'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ExtrusionPentagon.x3d' name='identifier'/>
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <NavigationInfo DEF='ExamineMode' type='"EXAMINE" "ANY"'/>
    <NavigationInfo DEF='FlyPanNavigation' type='"FLY" "ANY"'/>
    <Background skyColor='1 1 1'/>
    <Viewpoint description='ExtrusionPentagon' orientation='-1 0 0 0.2' position='0 3.17 12.85'/>
    <Viewpoint description='Oblique view from above' orientation='-1 0.015 -0.006 0.78' position='0.25 8.98 8.99'/>
    <Viewpoint description='Overhead view' orientation='1 0 0 -1.57' position='0 13.15 0'/>
    <Background groundColor='0.9 0.9 0.9' skyColor='0.9 0.9 0.9'/>
    <Transform translation='-4 0 0'>
      <Shape>
        <!-- be sure to order cross-section points so that normal is upward -->
        <Extrusion crossSection='-3.5 -1 -2.1 2.9 2.2 2.9 3.6 -1 0 -3.5 -3.5 -1' scale='1 1 1 1 0.2 0.2' solid='true' spine='0 0 0 0 1 0 0 1.6 0'/>
        <Appearance>
          <Material diffuseColor='0.2 0.6 0.5'/>
        </Appearance>
      </Shape>
    </Transform>
    <!-- Utilize ExtrusionCrossSection ProtoInstance as a Shape node (with containerField="children")
         rather than an Extrusion node (with containerField="geometry"). -->
    <ExternProtoDeclare>
    <Transform translation='4 0 0'>
      <ProtoInstance DEF='PentagonExtrusion' name='ExtrusionCrossSection'>
        <fieldValue name='name' value='PentagonExtrusion'/>
        <fieldValue name='crossSection' value='-3.5 -1 -2.1 2.9 2.2 2.9 3.6 -1 0 -3.5 -3.5 -1'/>
        <fieldValue name='spine' value='0 0 0 0 1 0 0 1.6 0'/>
        <fieldValue name='scale' value='1 1 1 1 0.2 0.2'/>
        <fieldValue name='lineColor' value='0.7 0.7 0'/>
        <fieldValue name='ccw' value='true'/>
        <fieldValue name='crossSectionMaterial'>
          <Material diffuseColor='0.6 0.6 0.2'/>
        </fieldValue>
        <fieldValue name='extrusionMaterial'>
          <Material diffuseColor='0.3 0.6 0.6' transparency='0.6'/>
        </fieldValue>
        <fieldValue name='traceEnabled' value='false'/>
      </ProtoInstance>
    </Transform>
  </Scene>
</X3D>
```



27:19 INS

X3D-Edit user interface for Extrusion

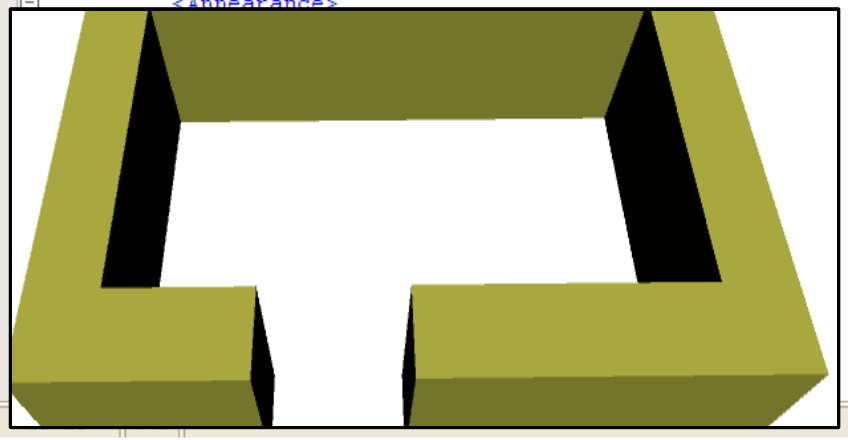




ExtrusionRoomWalls.x3d - Editor

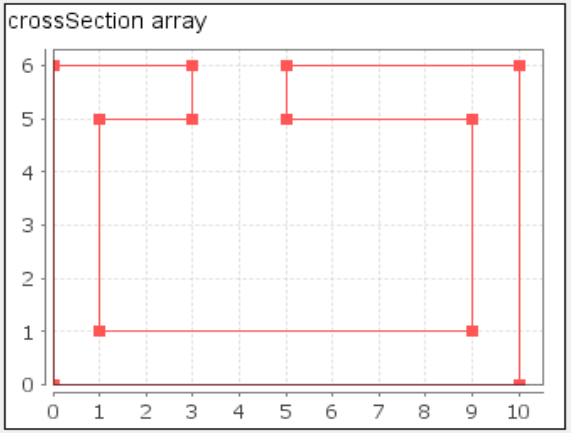
ExtrusionRoomWalls.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1'   xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
                    xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.1.xsd'>
  <head>
    <meta content='ExtrusionRoomWalls.x3d' name='title'/>
    <meta content='Wall definition for a room, defined as a cross section and extruded vertically.' name='description'/>
    <meta content='Don Brutzman' name='creator'/>
    <meta content='Naval Postgraduate School' name='organization'/>
    <meta content='3 September 2005' name='created'/>
    <meta content='3 September 2005' name='modified'/>
    <meta content='Extrusion pentagon' name='subject'/>
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ExtrusionRoomWalls.x3d'
          name='identifier'/>
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <Viewpoint description='ExtrusionRoomWalls' orientation='1 0 0 -0.2' position='5 4 15'/>
    <Viewpoint description='Oblique view from above' orientation='1 0 0 -0.78' position='5 10 13'/>
    <Viewpoint description='Overhead view' orientation='1 0 0 -1.57' position='5 12 3'/>
    <Background skyColor='1 1 1'/>
    <Transform>
      <Shape>
        <!-- be sure to order cross-section points so that normal is upward -->
        <Extrusion crossSection='0 0 0 6 3 6 3 5 1 5 1 1 9 1 9 5 5 5 5 6 10 6 10 0'
                   convex='false' solid='true' spine='0 0 0 0 2.5 0'/>
        <Appearance>
```



crossSection array

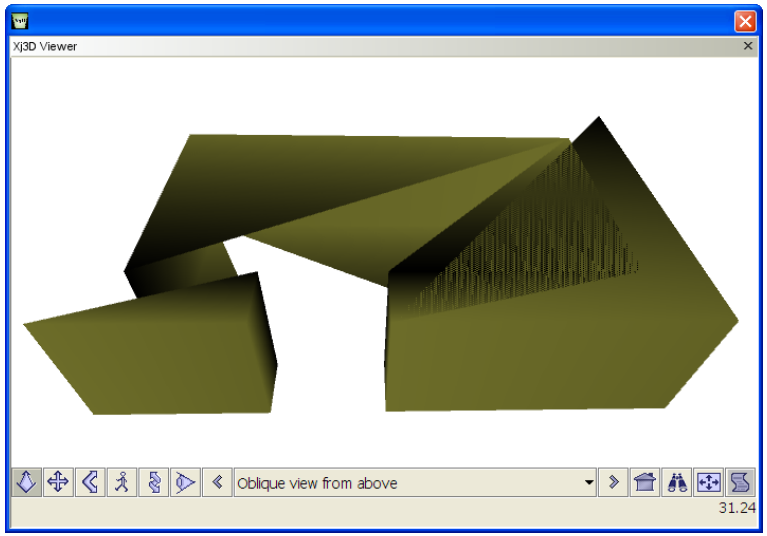
	x	y
0	0	0
1	0	6
2	3	6
3	3	5
4	1	5
5	1	1
6	9	1
7	9	5
8	5	5
9	5	6
10	10	6
11	10	0
12	0	0



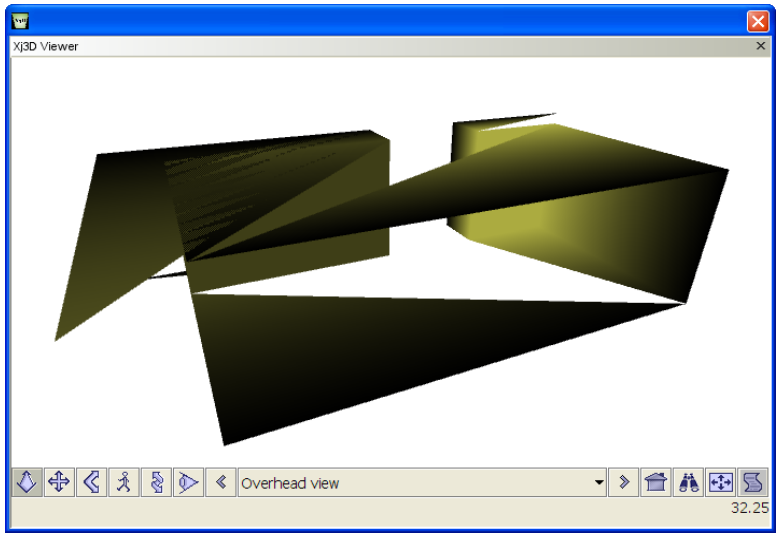
Concave geometry defects

Concave geometry with *convex*='true' can lead to confused geometry results

- Nonsensical polygons
- Aliasing (tearing) of coplanar polygons
- To correct: set *convex*='false'



Erroneous rendering



Erroneous rendering

Debugging Extrusion problems 1

Unlike most other nodes, ill-defined geometry is possible with Extrusion. Things to check:

- Verify proper *crossSection*, *spine*, *scale*, *orientation* array values and lengths
- Set *convex*='false' if geometry might be concave
- Set *solid*='false' to render inside and outside, eliminating "invisible geometry" when viewed from behind or inside the exterior hull
- Set *ccw*='false' if *crossSection* might be defined in clockwise direction

Debugging Extrusion problems 2

Counting checks

- Length of 2-tuple *scale* array must be 0, 1, or match length of 3-tuple *spine* array
- Length of 4-tuple *orientation* array must be 0, 1, or match length of 3-tuple *spine* array
- Values in *scale* and *crossSection* arrays must be multiple of 2 (MFVec2f)
- Values in *spine* array must be multiple of 3 (MFVec3f)
- Values in *orientation* array must be multiple of 4 (MFRotation)

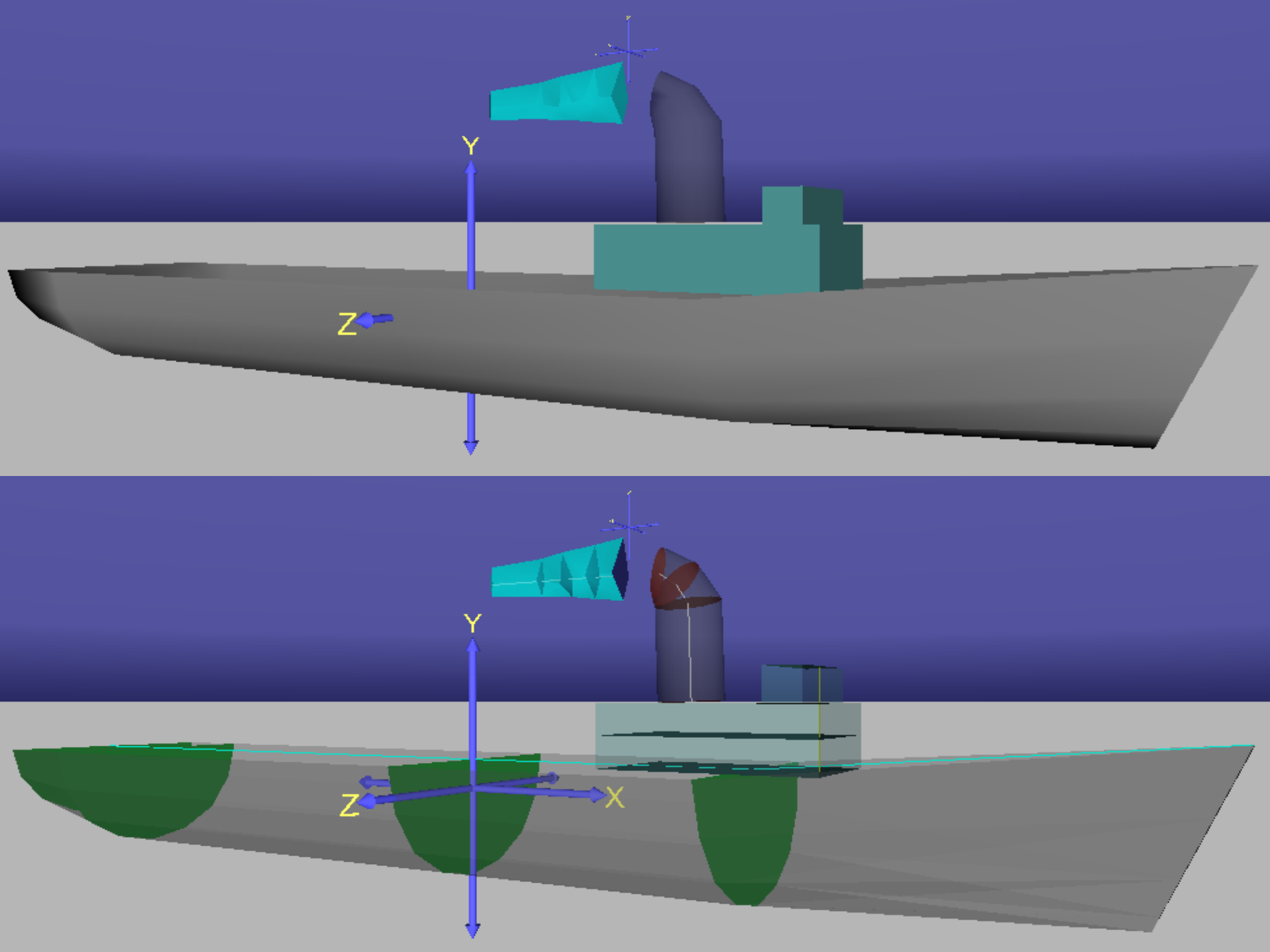
ExtrusionCrossSection prototype

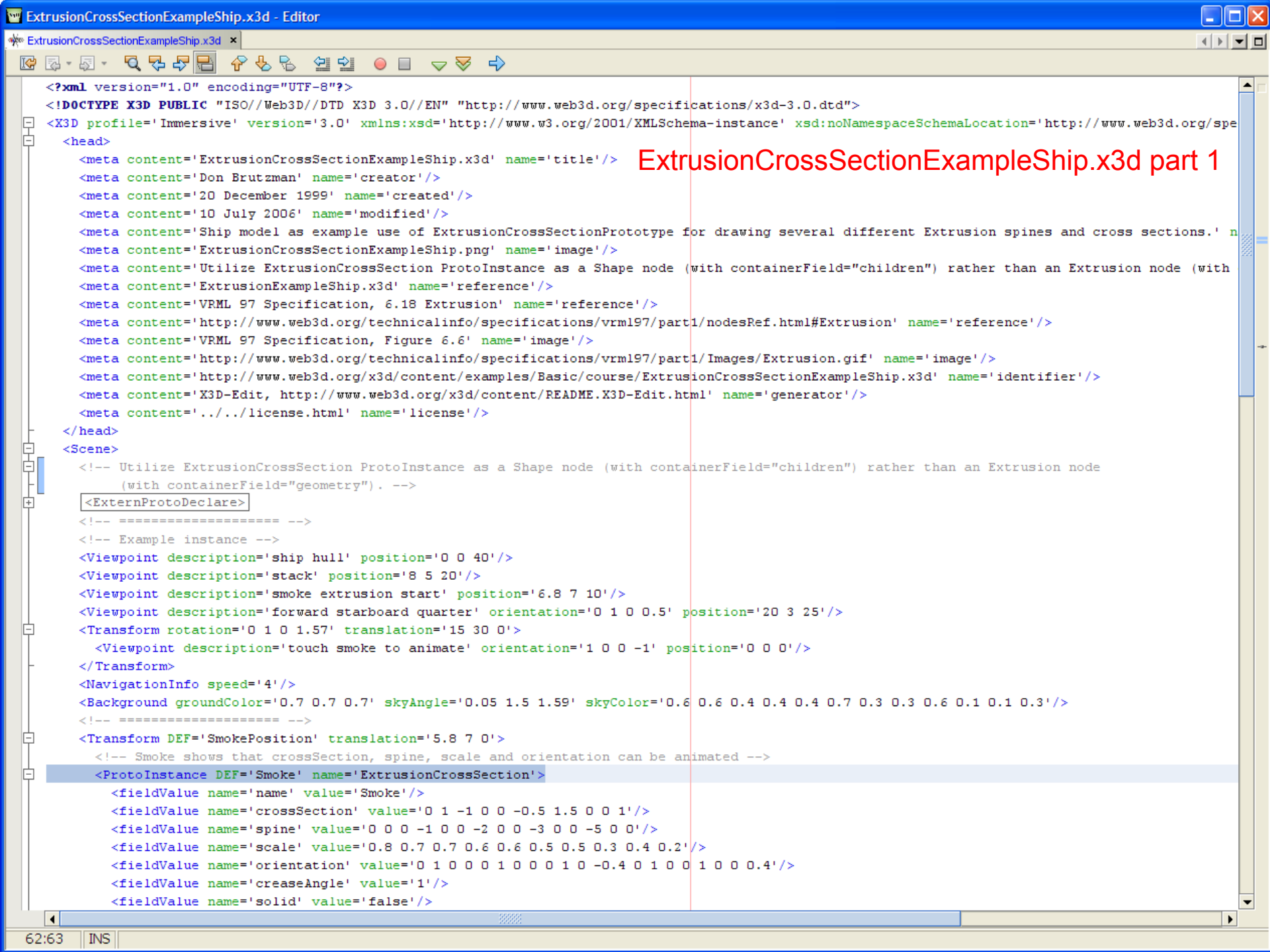
Preceding ExtrusionPentagon.x3d example scene contains a new construct: a Prototype node

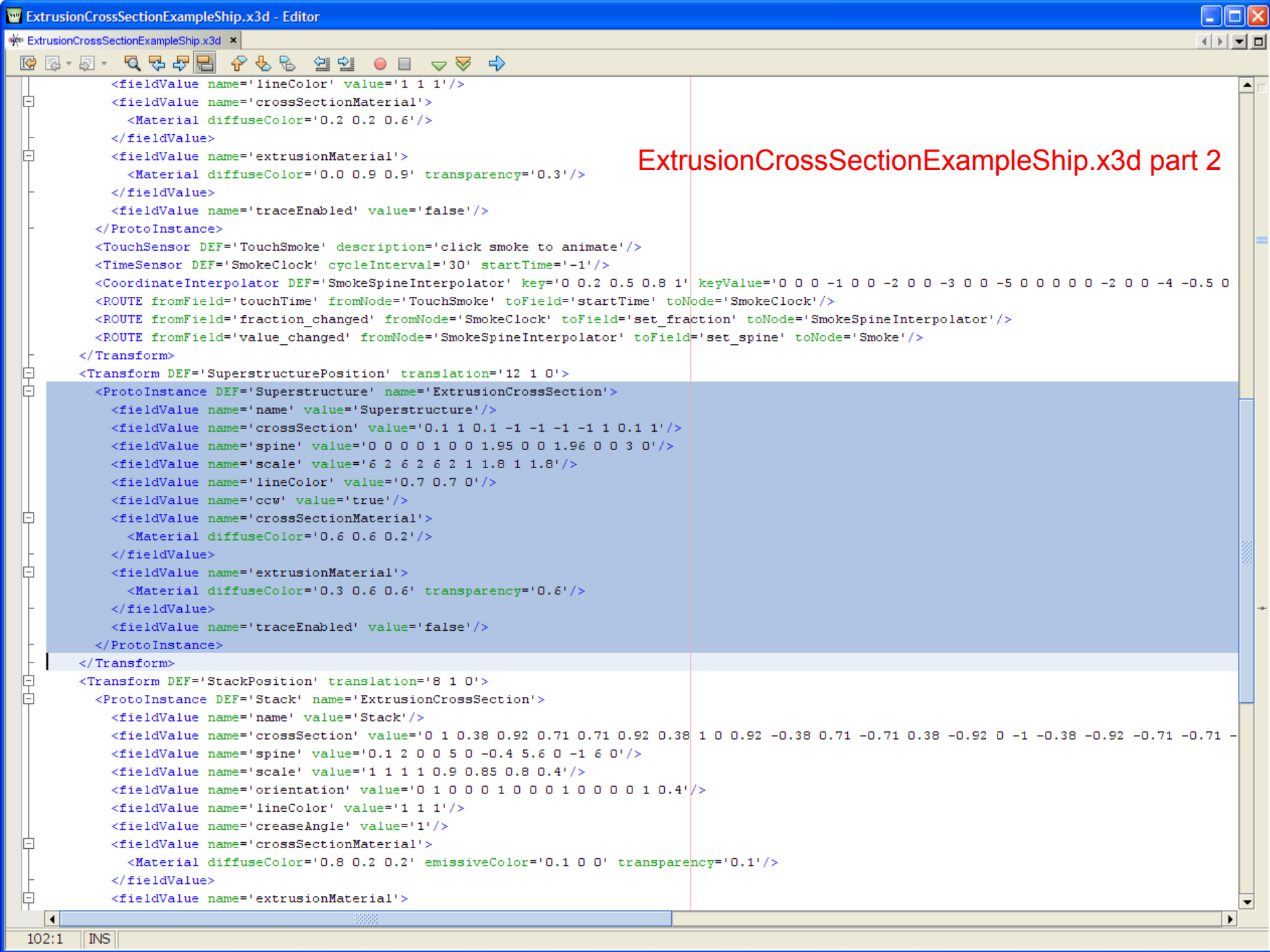
- ExternProtoDeclare refers to external prototype url and defines field signatures
- ProtoInstance creates an instance of the new node
- fieldValue definitions provide parameter values, in this case the same values as Extrusion of interest

Result is a specially computed Extrusion showing *crossSection* planes, *spine*, transparent sides

- Can provide helpful insight and debugging support


Y
Z
Y
Z
X


ExtrusionCrossSectionExampleShip.x3d - Editor
ExtrusionCrossSectionExampleShip.x3d
ExtrusionCrossSectionExampleShip.x3d part 1
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile='Immersive' version='3.0' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/spe
<head>
<meta content='ExtrusionCrossSectionExampleShip.x3d' name='title'/>
<meta content='Don Brutzman' name='creator'/>
<meta content='20 December 1999' name='created'/>
<meta content='10 July 2006' name='modified'/>
<meta content='Ship model as example use of ExtrusionCrossSectionPrototype for drawing several different Extrusion spines and cross sections.' n
<meta content='ExtrusionCrossSectionExampleShip.png' name='image'/>
<meta content='Utilize ExtrusionCrossSection ProtoInstance as a Shape node (with containerField="children") rather than an Extrusion node (with
<meta content='ExtrusionExampleShip.x3d' name='reference'/>
<meta content='VRML 97 Specification, 6.18 Extrusion' name='reference'/>
<meta content='http://www.web3d.org/technicalinfo/specifications/vrml97/part1/nodesRef.html#Extrusion' name='reference'/>
<meta content='VRML 97 Specification, Figure 6.6' name='image'/>
<meta content='http://www.web3d.org/technicalinfo/specifications/vrml97/part1/Images/Extrusion.gif' name='image'/>
<meta content='http://www.web3d.org/x3d/content/examples/Basic/course/ExtrusionCrossSectionExampleShip.x3d' name='identifier'/>
<meta content='X3D-Edit, http://www.web3d.org/x3d/content/README.X3D-Edit.html' name='generator'/>
<meta content='../../license.html' name='license'/>
</head>
<Scene>
<!-- Utilize ExtrusionCrossSection ProtoInstance as a Shape node (with containerField="children") rather than an Extrusion node
(with containerField="geometry"). -->
<ExternProtoDeclare>
<!-- ==================== -->
<!-- Example instance -->
<Viewpoint description='ship hull' position='0 0 40'/>
<Viewpoint description='stack' position='8 5 20'/>
<Viewpoint description='smoke extrusion start' position='6.8 7 10'/>
<Viewpoint description='forward starboard quarter' orientation='0 1 0 0.5' position='20 3 25'/>
<Transform rotation='0 1 0 1.57' translation='15 30 0'>
<Viewpoint description='touch smoke to animate' orientation='1 0 0 -1' position='0 0 0'/>
</Transform>
<NavigationInfo speed='4'/>
<Background groundColor='0.7 0.7 0.7' skyAngle='0.05 1.5 1.59' skyColor='0.6 0.6 0.4 0.4 0.4 0.7 0.3 0.3 0.6 0.1 0.1 0.3'/>
<!-- ==================== -->
<Transform DEF='SmokePosition' translation='5.8 7 0'>
<!-- Smoke shows that crossSection, spine, scale and orientation can be animated -->
<ProtoInstance DEF='Smoke' name='ExtrusionCrossSection'>
<fieldValue name='name' value='Smoke'/>
<fieldValue name='crossSection' value='0 1 -1 0 0 -0.5 1.5 0 0 1'/>
<fieldValue name='spine' value='0 0 0 -1 0 0 -2 0 0 -3 0 0 -5 0 0'/>
<fieldValue name='scale' value='0.8 0.7 0.7 0.6 0.6 0.5 0.5 0.3 0.4 0.2'/>
<fieldValue name='orientation' value='0 1 0 0 0 1 0 0 0 1 0 -0.4 0 1 0 0 1 0 0 0.4'/>
<fieldValue name='creaseAngle' value='1'/>
<fieldValue name='solid' value='false'/>
62:63
INS


ExtrusionCrossSectionExampleShip.x3d - Editor
ExtrusionCrossSectionExampleShip.x3d
ExtrusionCrossSectionExampleShip.x3d part 2
<fieldValue name='lineColor' value='1 1 1'/>
<fieldValue name='crossSectionMaterial'>
<Material diffuseColor='0.2 0.2 0.6'/>
</fieldValue>
<fieldValue name='extrusionMaterial'>
<Material diffuseColor='0.0 0.9 0.9' transparency='0.3'/>
</fieldValue>
<fieldValue name='traceEnabled' value='false'/>
</ProtoInstance>
<TouchSensor DEF='TouchSmoke' description='click smoke to animate'/>
<TimeSensor DEF='SmokeClock' cycleInterval='30' startTime='-1'/>
<CoordinateInterpolator DEF='SmokeSpineInterpolator' key='0 0.2 0.5 0.8 1' keyValue='0 0 0 -1 0 0 -2 0 0 -3 0 0 -5 0 0 0 0 0 -2 0 0 -4 -0.5 0
<ROUTE fromField='touchTime' fromNode='TouchSmoke' toField='startTime' toNode='SmokeClock'/>
<ROUTE fromField='fraction_changed' fromNode='SmokeClock' toField='set_fraction' toNode='SmokeSpineInterpolator'/>
<ROUTE fromField='value_changed' fromNode='SmokeSpineInterpolator' toField='set_spine' toNode='Smoke'/>
</Transform>
<Transform DEF='SuperstructurePosition' translation='12 1 0'>
<ProtoInstance DEF='Superstructure' name='ExtrusionCrossSection'>
<fieldValue name='name' value='Superstructure'/>
<fieldValue name='crossSection' value='0.1 1 0.1 -1 -1 -1 -1 1 0.1 1'/>
<fieldValue name='spine' value='0 0 0 0 1 0 0 1.95 0 0 1.96 0 0 3 0'/>
<fieldValue name='scale' value='6 2 6 2 6 2 1 1.8 1 1.8'/>
<fieldValue name='lineColor' value='0.7 0.7 0'/>
<fieldValue name='ccw' value='true'/>
<fieldValue name='crossSectionMaterial'>
<Material diffuseColor='0.6 0.6 0.2'/>
</fieldValue>
<fieldValue name='extrusionMaterial'>
<Material diffuseColor='0.3 0.6 0.6' transparency='0.6'/>
</fieldValue>
<fieldValue name='traceEnabled' value='false'/>
</ProtoInstance>
</Transform>
<Transform DEF='StackPosition' translation='8 1 0'>
<ProtoInstance DEF='Stack' name='ExtrusionCrossSection'>
<fieldValue name='name' value='Stack'/>
<fieldValue name='crossSection' value='0 1 0.38 0.92 0.71 0.71 0.92 0.38 1 0 0.92 -0.38 0.71 -0.71 0.38 -0.92 0 -1 -0.38 -0.92 -0.71 -0.71 -
<fieldValue name='spine' value='0.1 2 0 0 5 0 -0.4 5.6 0 -1 6 0'/>
<fieldValue name='scale' value='1 1 1 1 0.9 0.85 0.8 0.4'/>
<fieldValue name='orientation' value='0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0.4'/>
<fieldValue name='lineColor' value='1 1 1'/>
<fieldValue name='creaseAngle' value='1'/>
<fieldValue name='crossSectionMaterial'>
<Material diffuseColor='0.8 0.2 0.2' emissiveColor='0.1 0 0' transparency='0.1'/>
</fieldValue>
<fieldValue name='extrusionMaterial'>
102:1
INS

Extrusion	**Extrusion is a geometry node stretching a 2D cross section along a 3D-spine path in the local coordinate system Scaling/rotating cross-sections can produce a variety of shapes.** **Hint: insert a Shape node before adding geometry or Appearance.**
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!
spine	**[spine: accessType initializeOnly, type MFVec3f CDATA "0 0 0, 0 1 0"]** spine is a list of 3D points for a piecewise-linear curve forming a series of connected vertices, open or closed. This is the path along which the crossSection is extruded. **Hint:** number of spine points, scale values and orientation values must be the same.
crossSection	**[crossSection: accessType initializeOnly, type MFVec2f CDATA "1 1, 1 -1, -1 -1, -1 1, 1 1"]** An ordered set of 2D points drawing a piecewise-linear curve and forming a planar series of connected vertices. This provides a silhouette of the outer surface. **Warning:** match clockwise/counterclockwise or impossible/inverted geometry can result!
scale	**[scale: accessType initializeOnly, type MFVec2f CDATA "1 1"]** (0..infinity) scale is a list of 2D-scale parameters applied at each spine-aligned cross-section plane. **Hint:** number of spine points, scale values and orientation values must be the same. **Warning:** zero or negative scale values not allowed.
orientation	**[orientation: accessType initializeOnly, type MFRotation CDATA "0 0 1 0"]** orientation is a list of axis-angle orientation 4-tuples applied at each spine-aligned cross-section plane. **Hint:** number of spine points, scale values and orientation values must be the same.
beginCap	**[beginCap: accessType initializeOnly, type SFBool (true|false) "true"]** Whether beginning cap is drawn (similar to Cylinder top cap). **Warning:** cannot be changed after initial creation.
endCap	**[endCap: accessType initializeOnly, type SFBool (true|false) "true"]** Whether end cap is drawn (similar to Cylinder end cap). **Warning:** cannot be changed after initial creation.
ccw	**[ccw: accessType initializeOnly, type SFBool (true|false) "true"]** ccw = counterclockwise: ordering of vertex-coordinates orientation. **Hint:** ccw false can reverse solid (backface culling) and normal-vector orientation.
convex	**[convex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether all polygons in a shape are convex (true), or possibly concave (false). A convex polygon is planar, does not intersect itself, and has all interior angles < 180 degrees. **Warning:** concave geometry may be invisible default convex=true.

creaseAngle	**[creaseAngle: accessType initializeOnly, type SFFloat CDATA "0.0"]** [0..infinity) creaseAngle defines angle (in radians) where adjacent polygons are drawn with sharp edges or smooth shading. If angle between normals of two adjacent polygons is less than creaseAngle, smooth shading is rendered across the shared line segment. **Hint:** creaseAngle=0 means render all edges sharply, creaseAngle=3.14 means render all edges smoothly.
solid	**[solid: accessType initializeOnly, type SFBool (true|false) "true"]** Setting solid true means draw only one side of polygons (backface culling on), setting solid false means draw both sides of polygons (backface culling off). **Warning:** default value true can completely hide geometry if viewed from wrong side!
set_crossSection	**[set_crossSection: accessType inputOnly, type MFVec2f CDATA #FIXED ""]** An ordered set of 2D points drawing a piecewise-linear curve and forming a planar series of connected vertices. This provides a silhouette of the outer surface. **Warning:** match clockwise/counterclockwise or impossible/inverted geometry can result!
set_orientation	**[set_orientation: accessType inputOnly, type MFRotation CDATA #FIXED ""]** orientation is a list of axis-angle orientation 4-tuples applied at each spine-aligned cross-section plane. **Hint:** number of spine points, scale values and orientation values must be the same.
set_scale	**[set_scale: accessType inputOnly, type MFVec2f CDATA #FIXED ""]** (0..infinity) scale is a list of 2D-scale parameters applied at each spine-aligned cross-section plane. **Hint:** number of spine points, scale values and orientation values must be the same. **Warning:** zero or negative scale values not allowed.
set_spine	**[set_spine: accessType inputOnly, type MFVec3f CDATA #FIXED ""]** spine is a list of 3D points for a piecewise-linear curve forming a series of connected vertices, open or closed. This is the path along which the crossSection is extruded. **Hint:** number of spine points, scale values and orientation values must be the same.
containerField	**[containerField: NMTOKEN "geometry"]** 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.

Additional Resources

Geometry nodes

Chapter 2, Primitives

- Box, Cone, Cylinder, Sphere, Text / FontStyle

Chapter 6, Points Lines and Polygons

- PointSet, IndexedLineSet, IndexedFaceSet, ElevationGrid, Extrusion

We are here

Chapter 10, Geometry2D

- Arc2D,ArcClose2D, Circle2D, Disk2D, Polyline2D, Polypoint2D, Rectangle2D, TriangleSet2D

Chapter 13, Triangles and Quadrilaterals

- TriangleSet, TriangleStripSet, TriangleFanSet, QuadSet
- Both regular and Indexed versions

Advanced geometry nodes

Geospatial component

- GeoElevationGrid

NURBS component

- NurbsCurve, NurbsPatchSurface, NurbsSweptSurface, NurbsSwungSurface, NurbsTrimmedSurface

Programmable shaders component

- ComposedShader, PackagedShader, ProgramShader

Further information available in X3D Specification

- http://www.web3d.org/x3d/specifications

Scalable Vector Graphics (SVG)

SVG is an XML language for two-dimensional (2D) graphics and graphical applications

- World Wide Web Consortium (W3C) Recommendations, working group
- http://www.w3.org/Graphics/SVG

Because both X3D and SVG are written in XML, we've created an XSLT stylesheet that makes SVG plots of 2D data structures in X3D scenes

- X3dExtrusionToSvgViaXslt1.0.xslt
- Linked via pretty-print html, example follows

```
<?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 name='title' content='ExtrusionPentagon.x3d'/>
        <meta name='description' content='Simple regular pentagon extruded vertically.'/>
        <meta name='creator' content='Don Brutzman'/>
        <meta name='organization' content='Naval Postgraduate School'/>
        <meta name='created' content='3 September 2005'/>
        <meta name='modified' content='3 September 2005'/>
        <meta name='subject' content='Extrusion pentagon'/>
        <meta name='identifier' content=' http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ExtrusionPentagon.x3d '/>
        <meta name='generator' content='X3D-Edit, https://savage.nps.edu/X3D-Edit'/>
        <meta name='license' content='../license.html'/>
    </head>
    <!--
```

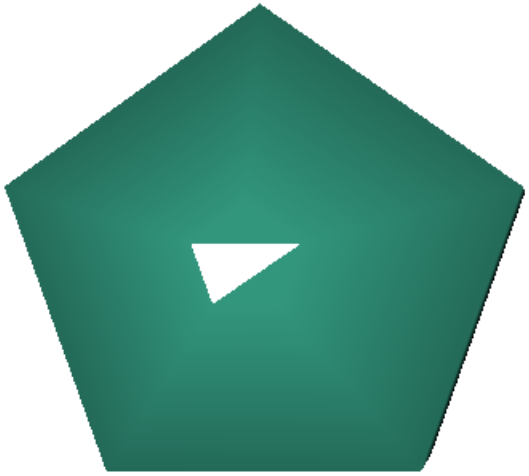
Index for ExternProtoDeclare definition: ExtrusionCrossSection
Index for DEF nodes: ExamineMode, FlyPanNavigation, PentagonExtrusion
Index for Viewpoint images: Viewpoint_1, Viewpoint_2, Viewpoint_3

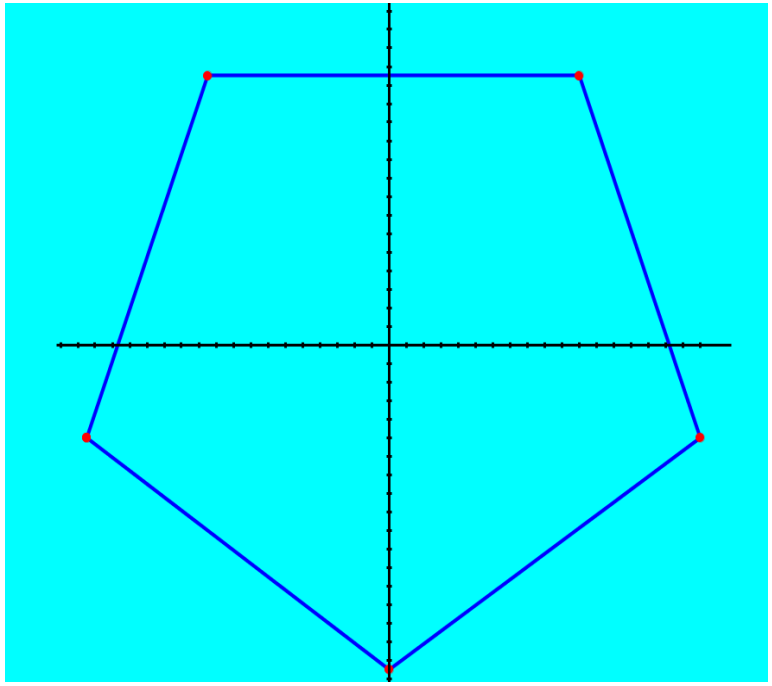
Select *description* to see .png image of viewpoint

```
    -->
    <Scene>
        <NavigationInfo DEF='ExamineMode' type='"EXAMINE" "ANY"' transitionType='"ANIMATE"' transitionTime='1.0'/>
        <NavigationInfo DEF='FlyPanNavigation' type='"FLY" "ANY"' transitionType='"ANIMATE"' transitionTime='1.0'/>
        <Background skyColor='1 1 1'/>
        <Viewpoint description='ExtrusionPentagon' orientation='-1 0 0 0.2' position='0 3.17 12.85'/>
        <Viewpoint description='Oblique view from above' orientation='-1 0.015 -0.006 0.78' position='0.25 8.98 8.99'/>
        <Viewpoint description='Overhead view' orientation='1 0 0 -1.57' position='0 13.15 0'/>
        <Background groundColor='0.9 0.9 0.9' skyColor='0.9 0.9 0.9'/>
        <Transform translation='-4 0 0'>
            <Shape>
                <!-- be sure to order cross-section points so that normal is upward -->
                <Extrusion crossSection='-3.5 -1 -2.1 2.9 2.2 2.9 3.6 -1 0 -3.5 -3.5 -1' scale='1 1 1 1 0.2 0.2' spine='0 0 0 0 1 0 0 1.6 0'/>
                <Appearance>
                    <Material diffuseColor='0.2 0.6 0.5'/>
                </Appearance>
            </Shape>
```

Select *crossSection* to see .svg outline

Viewpoint .png and Extrusion .svg output images





TODO: fix clipping artifacts at top of pentagons

Chapter Summary

Chapter Summary 1

Polygonal geometry is essence of X3D graphics

- Also for most other computer graphics approaches

The rendering of almost every geometric shape is usually based on tessellation into triangles

Working with examples is the best way to learn.

Be patient, the principles are consistent, and practice helps make principles familiar.

Chapter Summary 2

Triangles, single-sided polygons, normal vectors

Common fields: *ccw*, *convex*, *creaseAngle*, etc.

Geometry nodes, part 2

- Color and ColorRGBA
- Coordinate and CoordinateDouble
- PointSet
- IndexedLineSet and LineSet
- IndexedFaceSet
- ElevationGrid
- Extrusion

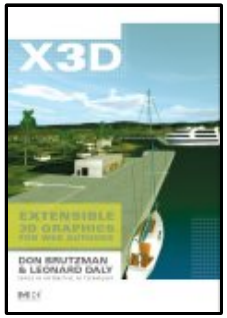
Suggested exercises

- Produce a simple graph of any sampled or functional X-Y data using IndexedLineSet
  - Also include axes and min/mix scale labels
- Write a simple program (in any language) that outputs coordinates for a circle's circumference
  - Insert outputs into a PointSet node for display
  - Show change when points are more closely spaced
- Build a simple object using an IndexedFaceSet
  - Add other IFS nodes with different Material values
- Build examples with ElevationGrid, Extrusion

References

References 1

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



- Chapter 6, Geometry 2: Points Lines and Polygons
- http://x3dGraphics.com
- http://x3dgraphics.com/examples/X3dForWebAuthors

X3D Resources

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

References 2

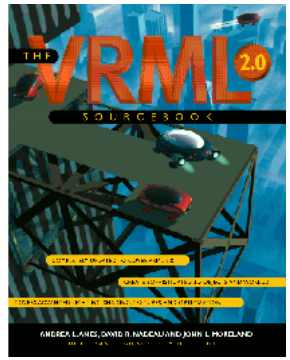
X3D Scene Authoring Hints

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

X3D Graphics Specification

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

References 3



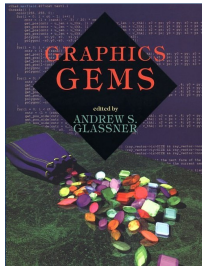
VRML 2.0 Sourcebook by Andrea L. Ames, David R. Nadeau, and John L. Moreland, John Wiley & Sons, 1996.

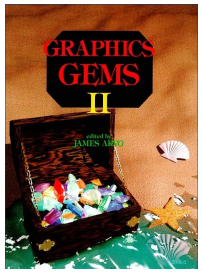
- http://www.wiley.com/legacy/compbooks/vrml2sbk/cover/cover.htm
- http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook
- Chapter 13 – Points Lines Faces
- Chapter 14 – Elevation Grid
- Chapter 15 – Extrusion
- Chapter 16 – Color

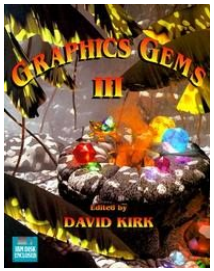
References 4

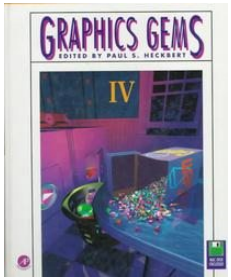
Graphics Gems book series: many algorithms

- http://www.graphicsgems.org









Journal of Graphics Tools (jgt): many algorithms

- http://jgt.akpeters.com

References 5

Point-Based Graphics, Markus Gross and Hanspeter Pfister, editors, Morgan Kaufmann Publishers, 2007.

- http://www.elsevier.com/wps/find/bookdescription.cws_home/710117/description#description

Point-based Graphics Resources

- Ke-Sen Huang
- http://kesen.huang.googlepages.com/PointBasedPaper.html

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

web|3D CONSORTIUM

CGEMS, SIGGRAPH, Eurographics

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

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

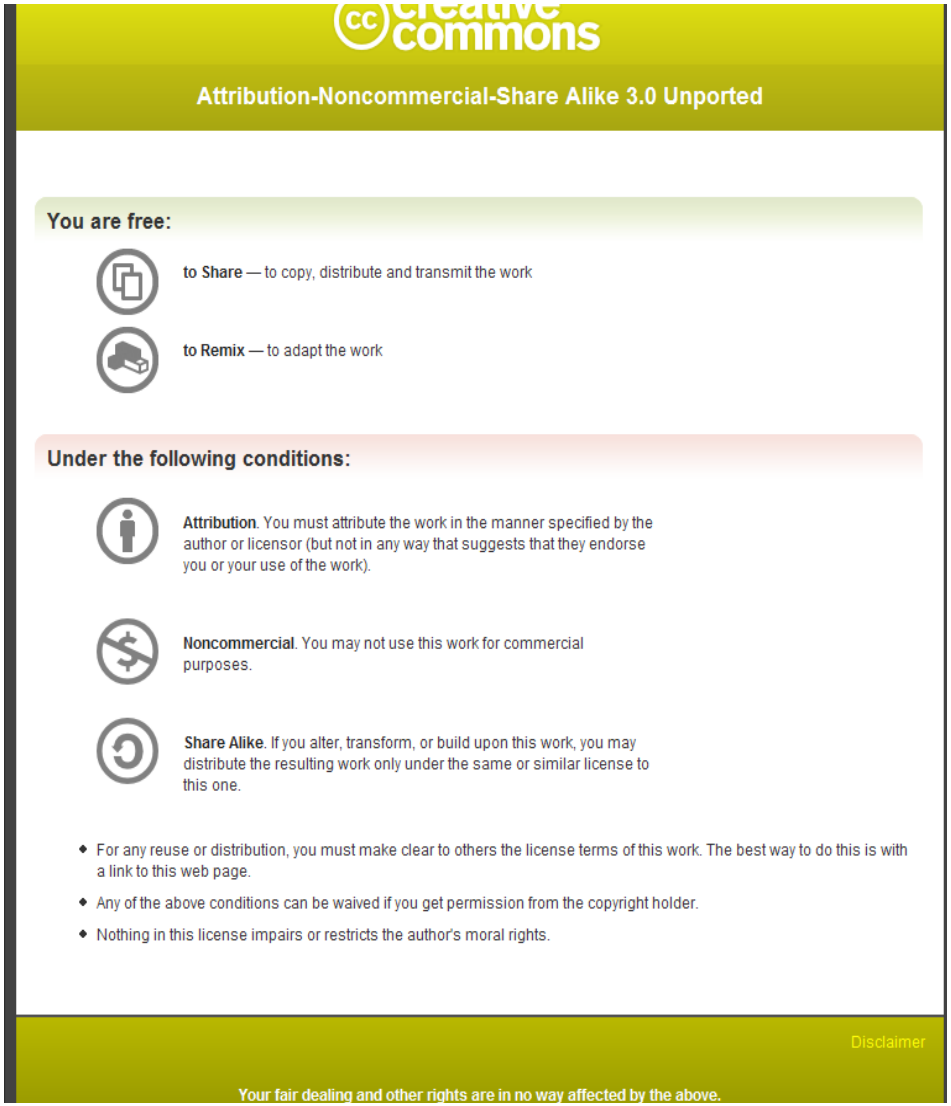
X3D for Web Authors recognized by CGEMS! ☺

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

CGEMS supported by SIGGRAPH, Eurographics

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

X3D Graphics for Web Authors

Chapter 6

Points, Lines and Polygons

Drawing is a struggle between nature and the artist, in which the better the artist understands the intentions of nature, the more easily he will triumph over it. For him it is not a question of copying, but of interpreting in a simpler and more luminous language.

Charles Baudelaire, *On the Ideal and the Model,* 1846.

web3D CONSORTIUM

1

Contents

web|3D CONSORTIUM

2

Chapter Overview

3

Many different geometry nodes

An excellent aspect of X3D is that there are many different ways to create geometry

- Chapter 2, Geometry Primitives
- Chapter 6, Points, Lines and Polygons
- Chapter 10, Geometry2D Nodes
- Chapter 13, Triangles and Quadrilaterals

These are all handled consistently inside a Shape node with corresponding Appearance

web3D CONSORTIUM

4

The primitive geometry nodes are *tessellated* (turned into triangles) by X3D browsers.

The primitive geometry nodes might also be written using IndexedFaceSet representations. This would let authors control how many polygons are used, meaning that higher fidelity to curved surfaces and higher-resolution shapes can be produced than might be produced by a given browser.

Also of interest is the X3D Non-Uniform Rational B-Spline (NURBS) component.

Fundamental geometry nodes

Geometry nodes in this chapter include points, lines, and indexed face sets

These nodes fundamental and can represent almost any shape

- Tools can convert other geometry to simpler forms
- Thus most are part of Interchange profile for broadest possible usage and adaptability

Some browsers support viewing geometry in wireframe (line) or point (cloud) mode, which can help to reveal internal geometric structure

- http://en.wikipedia.org/wiki/Wire-frame_model

web3D CONSORTIUM

5

Tip: Xj3D viewer in X3D-Edit includes wireframe (line) and point rendering modes

- Alt-shift-D on selected Xj3D window to unDock (or Dock) it from the X3D-Edit frame
- Alt-shift-W toggle Wireframe rendering
- Alt-shift-P toggle Point rendering

If the Xj3D window is undocked, it can be expanded to full screen.

There are no processing commands in X3D to change browser rendering styles from polygonal to wireframe (line) or point (cloud) modes. Ordinarily this is only directable by the user, if the browser offers a user interface to change the rendering mode.

- Sometimes a browser offers a custom API that allows a programmer to control this.

With a little care during design, it is possible for authors to re-use Coordinate point sets to be used either for IndexedFaceSet and IndexedLineSet nodes,

Overview: Points, Lines and Polygons

Triangles, single-sided polygons, normal vectors

Common fields: *ccw*, *convex*, *creaseAngle*, etc.

Geometry nodes, part 2:

- Coordinate and CoordinateDouble
- Color and ColorRGBA
- PointSet
- IndexedLineSet and LineSet
- IndexedFaceSet
- ElevationGrid
- Extrusion

web3D CONSORTIUM

6

back to Table of Contents

Concepts

These concepts are also common to Chapter 13, Geometry part 4: Triangles and Quadrilaterals

Triangles

Triangles are the primary low-level geometry construct used by graphics software, hardware

- More complex shapes are reduced to triangles by the rendering software (known as tesselation)
- A triangle is always planar, allowing the material appearance to fill it

Sometimes quadrilaterals are used, but problem is that values might be non-coplanar due to roundoff (or authoring) error

- Which means that filling in material is ill defined, and not properly or repeatably renderable

web3D CONSORTIUM

<X3D>

8

Graphics hardware is highly optimized to favor triangles, performing something very simple, many times, extremely quickly.

Single-sided polygons

- Graphics engines always prefer simplicity in order to achieve maximum run-time performance
  - Top 3 considerations for graphics hardware: performance, performance, performance!
- Single-sided polygons take about half the time to draw than double-sided polygons
  - So if authors can arrange geometry so that only one side is ever visible to user, can go single-sided
  - Technical term: *backface culling*
  - Efficiency is rationale for many X3D default values
  - Example: default setting is *solid*='true'
  - Debugging hint: set *solid*='false' to show both sides

Important technique: set *solid*='false' to show both sides of all polygons, helping to expose inside-out or missing triangles. The performance handicap is typically slight, especially compared to the alternative: missing model parts!

Despite the historical preoccupation with hardware performance in 3D graphics, your time and end-user performance is even more important. Since graphics cards are becoming so fast that it is growing ever harder to overwhelm, this focus on supporting author and end-user efficiency is an important strength of X3D.

Review from Chapter 2

Common field: *solid*

In 3D graphics, all triangles have 2 sides

- Graphics term: backface culling only draws front sides

The *solid* field defines whether a geometry node has an inside or not, with a default value of true

- *solid*='true' means do not render (draw) the inside
- *solid*='false' means render both inside and outside

This approach reduces the number of polygons needing to be drawn, thus improving performance

Confusing if user gets lost inside invisible geometry

- Hint: set *solid*='false' to draw both sides

web3D CONSORTIUM

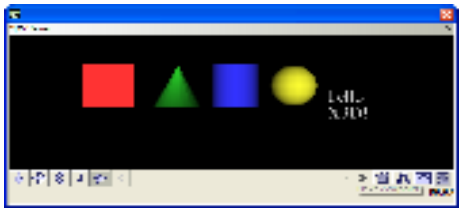
10

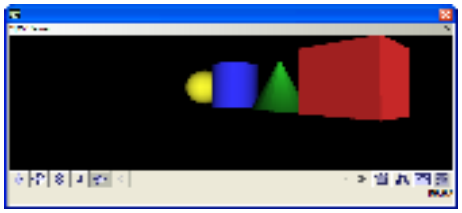
Review from Chapter 2

Common field: *solid*

To see an example of 'solid' geometry, rotate the GeometryPrimitives.x3d scene by 180 degrees

- Once rotated, the first four shapes remain visible, but the Text node disappears
- This is because *solid='true'* by default, so the reverse side of text is not drawn by default





Here we are still using the example

http://www.x3dbook.com/examples/X3dForWebAuthors/Chapter02-GeometryPrimitives/GeometryPrimitiveNodes.x3d

The original scene (on the left) is rotated about 150 degrees to the right. To do this, click on the left center of the screen and drag the mouse (pointer) to the right.

You need to have browser navigation in EXAMINE mode for this view rotation to work. In Xj3D, which is used in X3D-Edit, the EXAMINE mode icon is the stylized eye (fifth button on the lower left as shown here).

Normal vectors

The *normal* vector is perpendicular to the face, pointing away from the centroid of polygon

Direction of normal vector defined by order of points defining the polygon and right-hand rule

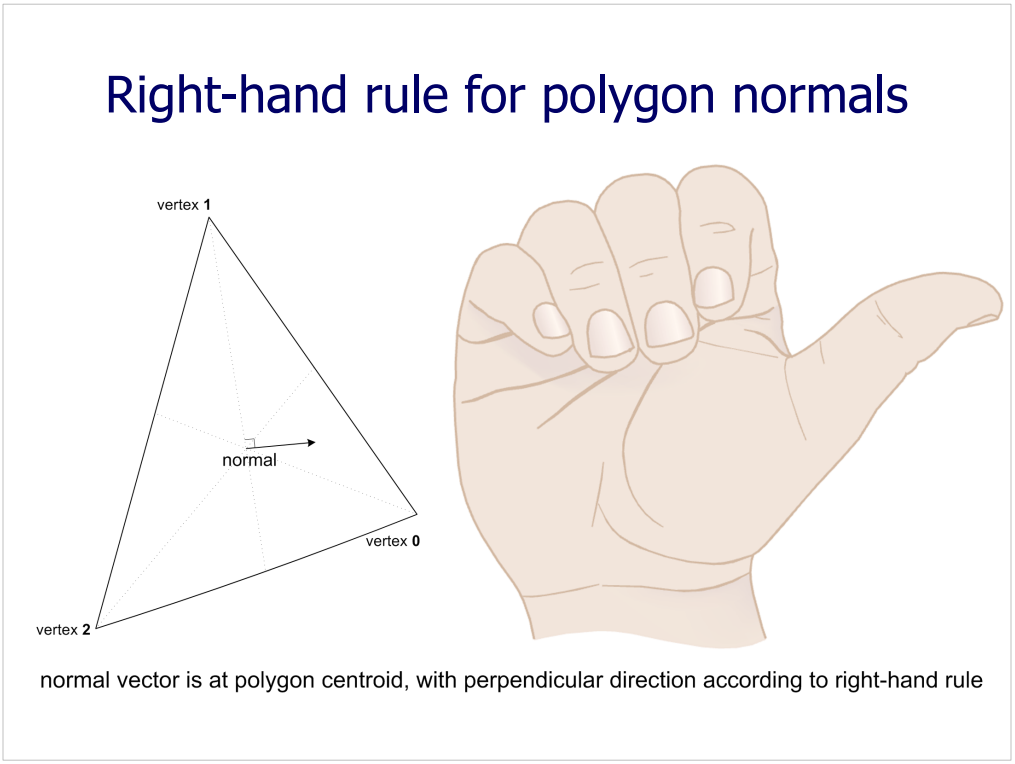
- Align curvature of fingers to match polygon vertex points in order: indices 0, 1, 2 ...
- Thumb points in direction of (positive) normal, which is the front-facing side
- Negative normal thus points in direction of backface

web3D CONSORTIUM

12

Normal vectors are only pertinent to polygons, not polyline segments or point sets. The Normal node is covered in Chapter 13, Triangles and Quadrilaterals.

The NormalInterpolator node is covered in Chapter 7, Event Animation.



This rule also applies to quadrilaterals and polygons that have more vertices.

Vertices that are not coplanar are degenerate, and can lead to erroneous normal computation and unpredictable rendering results.

Normal vectors are only pertinent to polygons, not polyline segments or point sets. The Normal node is covered in Chapter 13, Triangles and Quadrilaterals.

Note that order of vertices must match the curl of the right hand. If not, you are looking at the triangle backface and pointing in the opposite direction, rather than the (positive) normal direction.

Tool note: if you export an X3D (or VRML) model from another tool such as Maya or 3DSMax, you have the option to simply delete all of the autogenerated normals. X3D players are able to quickly and correctly compute these automatically when loading geometric models.

Common field: *ccw*

ccw (counter clockwise) indicates whether default direction of polygon normals is counterclockwise (default) or clockwise

- *ccw*='true' is right-hand rule
- *ccw*='false' is opposite

Hint: can correct some opposite-rendering geometry by reversing ccw value, rather than reordering all coordinates or indices

- Saves time on some import conversions

web|3D CONSORTIUM

14

Look at a circular clock face, and apply the right-hand rule at the center with the thumb pointing out from the wall. The curvature of the fingers is counterclockwise.

ccw type is SFBool (boolean).

Common geometry node patterns

```
<IndexedFaceSet>
  <Coordinate/>
  <Color/>
  <Normal/>
</IndexedFaceSet>
```

```
<IndexedLineSet>
  <Coordinate/>
  <Color/>
</IndexedLineSet>
```

etc.

web|3D CONSORTIUM

Common field: *colorPerVertex*

colorPerVertex indicates whether contained color values are applied to each vertex point (default), or to each polygonal face

- *colorPerVertex*='true' requires that # colors must equal # points
- *colorPerVertex*='false' requires that # colors must equal # polygons

IndexedFaceSet *colorPerVertex* true, false

IndexedLineSet *colorPerVertex* true, false

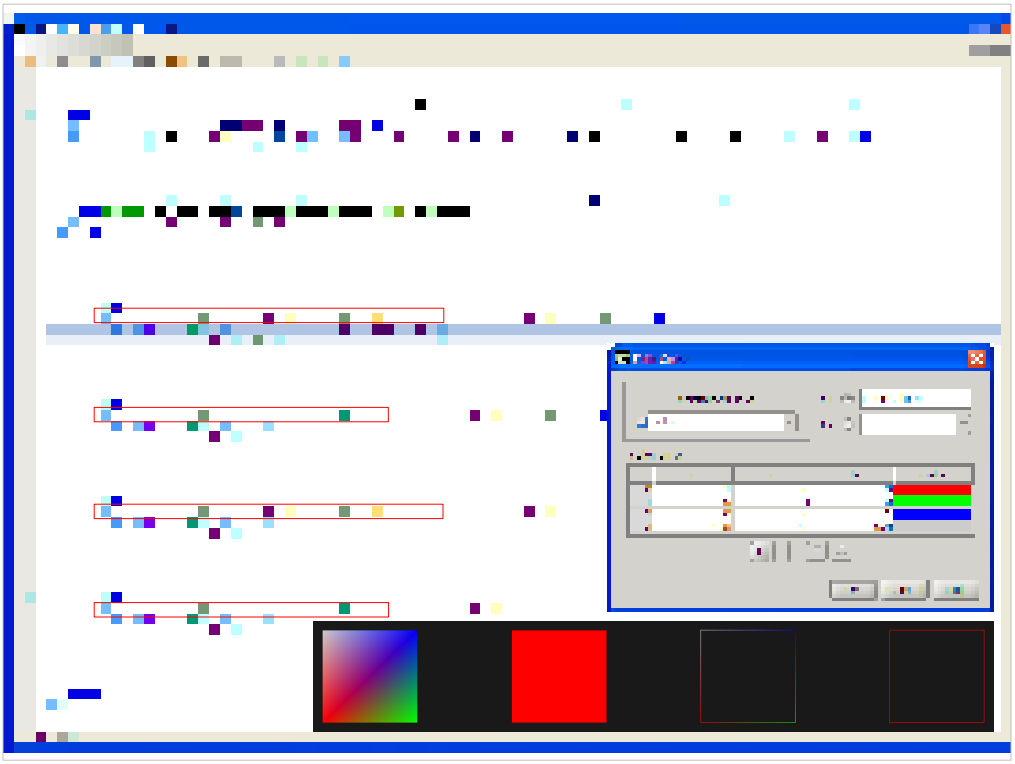
In this context, read # as 'number of'

colorPerVertex type is SFBool (boolean).

Given either value for *colorPerVertex:* counting the number of colors, along with the number of points or polygons, then making sure that those numbers are consistent, is an important correctness check.

X3D for Web Authors, Figure 6.1, p. 160

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ColorPerVertexExamples.x3d



http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ColorPerVertexExamples.x3d

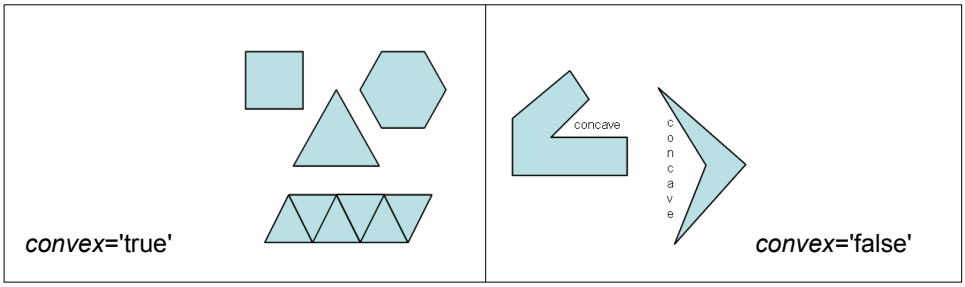
In addition to illustrating *colorPerVertex* effects, this example shows how Coordinate and Color nodes can be shared by both IndexedFaceSet and IndexedLineSet.

Also note how each set of *coordIndex* and *colorIndex* fields form a closed loop, both starting and ending with index 0 (prior to the end-of-polygon/end-of-polyline sentinel value -1).

Common field: *convex*

convex indicates whether an *n*-sided polygon has concave sides, meaning empty-space cavities

- *convex*='true' (default) means no concave sides
- *convex*='false' means concave sides may exist in the polygon, so extra care is needed to avoid hardware or software difficulty when rendering



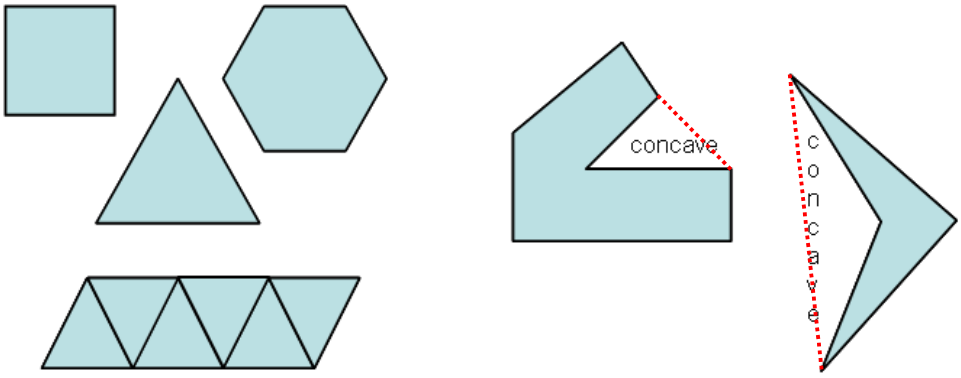
Software or hardware render engines will usually either re-triangulate concave geometry, or else apply more stringent rendering algorithms, in order to avoid mistakenly filling in the cavities found in a non-convex n-sided polygon.

convex type is SFBool (single-field Boolean, either true or false).

Note that triangles can only be convex, so triangulation always avoids concave problems.

X3D for Web Authors, Figure 6.2, p. 161

Visual test for concave geometry: whether a line segment drawn between any two of a polygon's vertices intersects space outside of the polygon.



Common field: *creaseAngle*

creaseAngle defines the angle (in radians) used to determine whether adjacent polygons are drawn with sharp edges or smooth shading

- If angle between polygons is less than *creaseAngle,* then smooth shading is used
- Smooth shading can conceal underlying tessellation

creaseAngle only affects shading within a single geometric shape, not exterior boundaries

- *creaseAngle*='0.0' means all edges are sharp
- *creaseAngle*='3.14159' means no edges are sharp

web3D CONSORTIUM

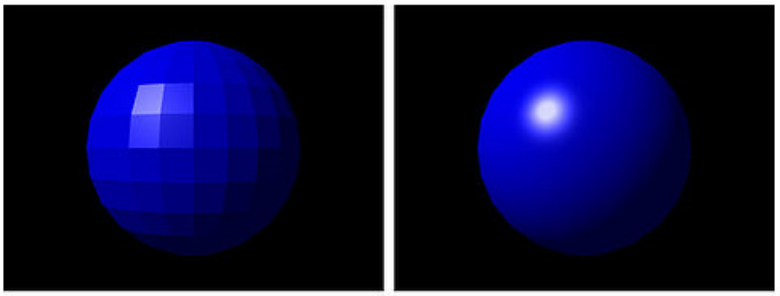
19

3.14159 = pi = 180 degrees. Note that it is not possible for adjacent polygons to have a relative angle > 180 degrees.

creaseAngle type is SFFloat (single-precision floating point).

Browsers perform *creaseAngle* computations by comparing normal vectors.

Many shapes look better with smooth shading. Do not hesitate to experiment with using *creaseAngle*='0' or another small (radian) value to improve visual realism.



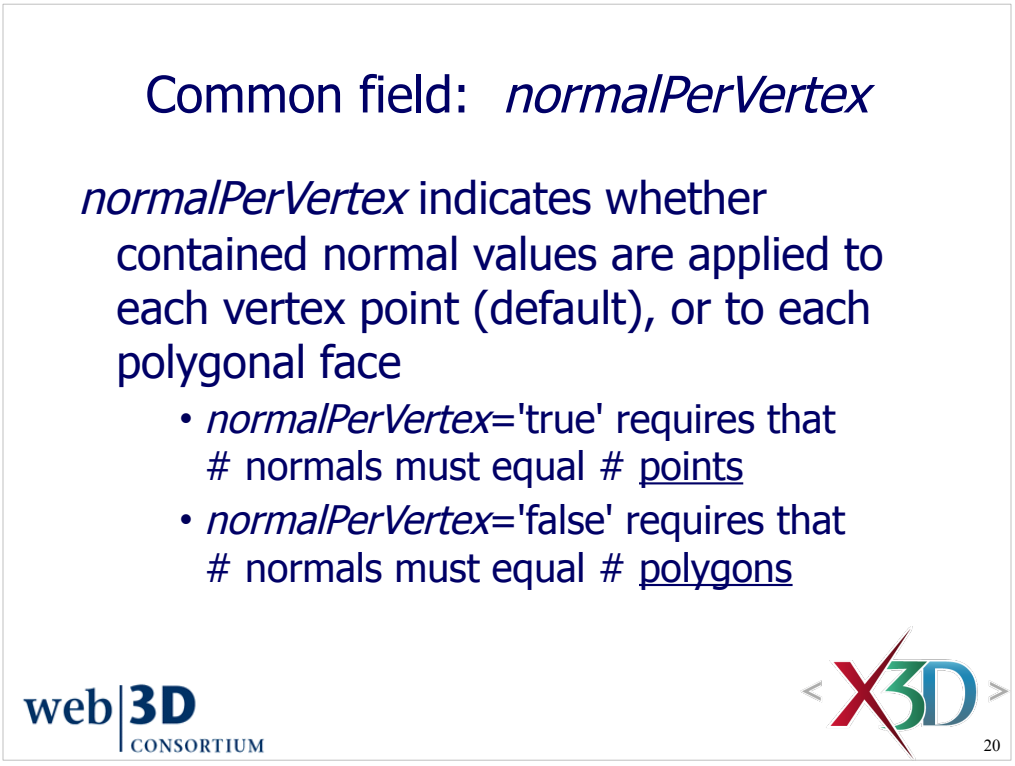
FLAT SHADING PHONG SHADING

The original (and still typically the most widely used) implementation algorithm is Phong shading, named for work by Bui Tuong Phong at University of Utah in 1973.

http://en.wikipedia.org/wiki/Phong_shading

http://en.wikipedia.org/wiki/Bui_Tuong_Phong

http://en.wikipedia.org/wiki/File:Phong-shading-sample.jpg



In this context, read # as 'number of'

normalPerVertex type is SFBool (boolean).

Given either value for *normalPerVertex:* counting the number of normals, along with the number of points or polygons, then making sure that those numbers are consistent, is an important correctness check.

Normal vectors are only pertinent to polygons, not polyline segments or point sets. The Normal node is covered in Chapter 13, Triangles and Quadrilaterals.

Bump mapping is a related advanced rendering technique that is used to vary normal vectors across a surface. Bump mapping techniques for X3D are not yet standardized (as of version 3.2).

- http://en.wikipedia.org/wiki/Bump_map

Common index fields: *coordIndex*, *colorIndex*, *normalIndex*

coordIndex, *colorIndex*, *normalIndex* each provide arrays of integer indices that connect individual vertices into polygons, then correlate corresponding Color/ColorRGBA, Coordinate or Normal values

- Initial index is 0
- Sentinel value -1 concludes polygon, polyline
- Maximum value equals (count - 1)
- Integer type MFInt32, default is empty array

Normal vectors are only pertinent to polygons, not polyline segments or point sets. The Normal node is covered in Chapter 13, Triangles and Quadrilaterals.

index counting checks

- *colorIndex* count must equal (**point** count - 1) when *colorPerVertex*='true', which is default
- *colorIndex* count must equal (**polygon** count - 1) when *colorPerVertex*='false'

- *normalIndex* count must equal (**point** count - 1) when *normalPerVertex*='true', which is default
- *normalIndex* count must equal (**polygon** count - 1) when *normalPerVertex*='false'

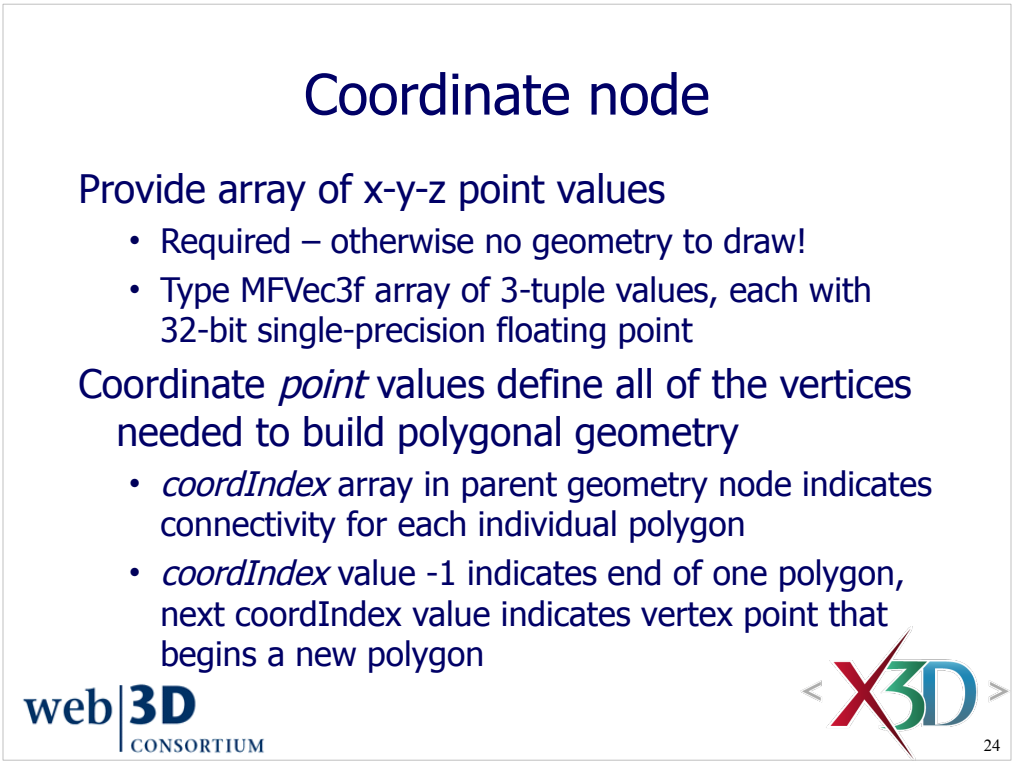
web|3D CONSORTIUM

22

Failure to meet these requirements is an error. Results are often unpredictable since the X3D Specification doesn't provide strict requirements for how to handle errors.

back to Table of Contents

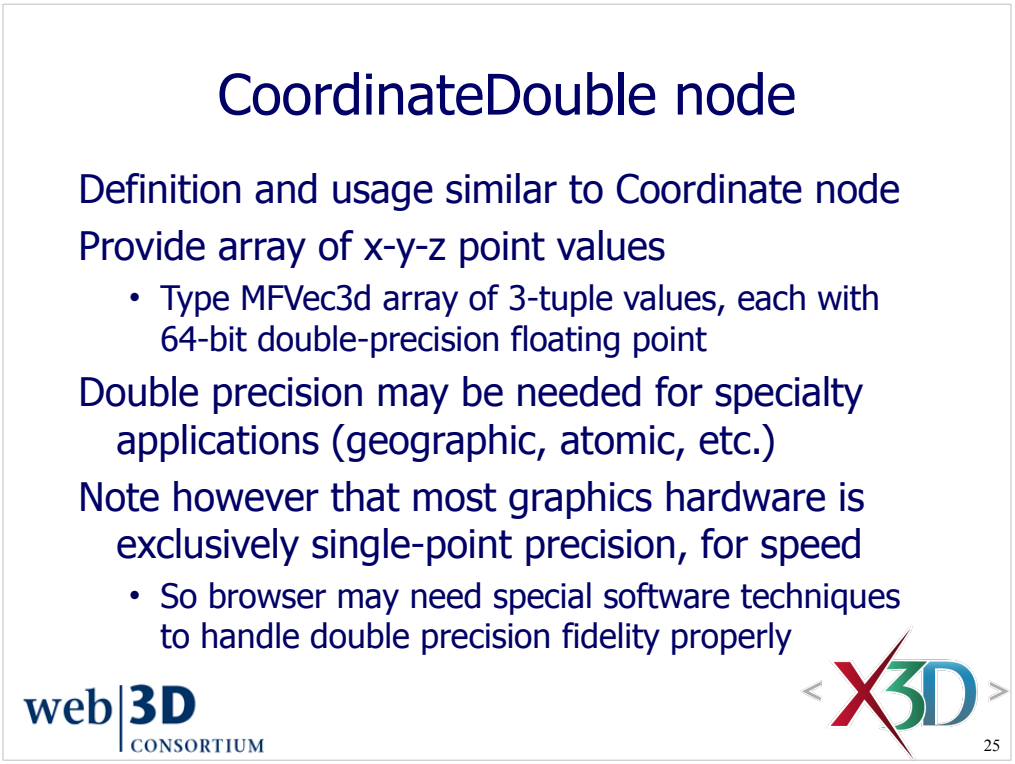
X3D Nodes and Examples



Commas are allowed as whitespace characters, but the X3D Schema will return an XML validation error if a comma appears within a given SFVec3f 3-tuple value.

X3D Canonicalization (C14N) reformatting removes intermediate comma characters.

Coordinate and CoordinateDouble nodes can appear as the coord field within the following nodes: PointSet, IndexedLineSet, IndexedFaceSet, IndexedTriangleFanSet, IndexedTriangleSet, IndexedTriangleStripSet, TriangleFanSet, TriangleSet, and TriangleStripSet.



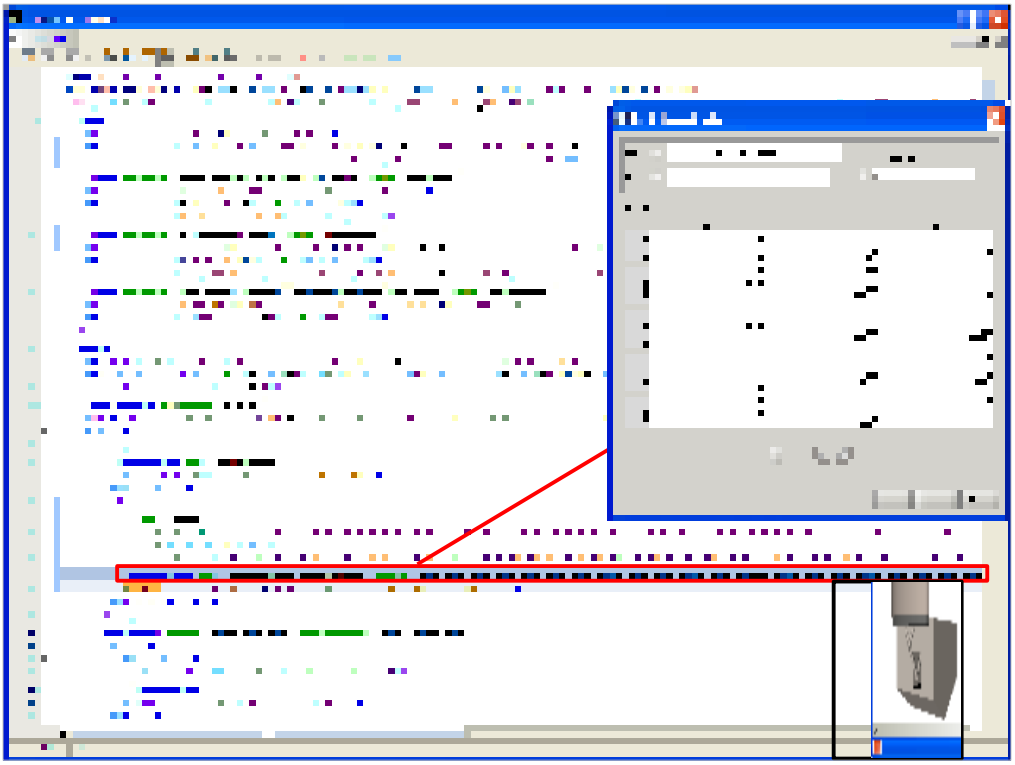
CoordinateDouble was not in VRML97. It was added to X3D to support advanced applications. It is not needed in most scenes, but is provided here for completeness.

Converting Coordinate nodes to CoordinateDouble nodes is usually a very bad idea, since almost all graphics hardware (and software) using single-precision floating point numbers. So don't do that!

Commas are allowed as whitespace characters, but the X3D Schema will return an XML validation error if a comma appears within a given SFVec3d 3-tuple value.

X3D Canonicalization (C14N) reformatting removes intermediate comma characters.

Coordinate and CoordinateDouble nodes can appear as the coord field within the following nodes: PointSet, IndexedLineSet, IndexedFaceSet, IndexedTriangleFanSet, IndexedTriangleSet, IndexedTriangleStripSet, TriangleFanSet, TriangleSet, and TriangleStripSet.



X3D for Web Authors, Figure 6.3, p. 164

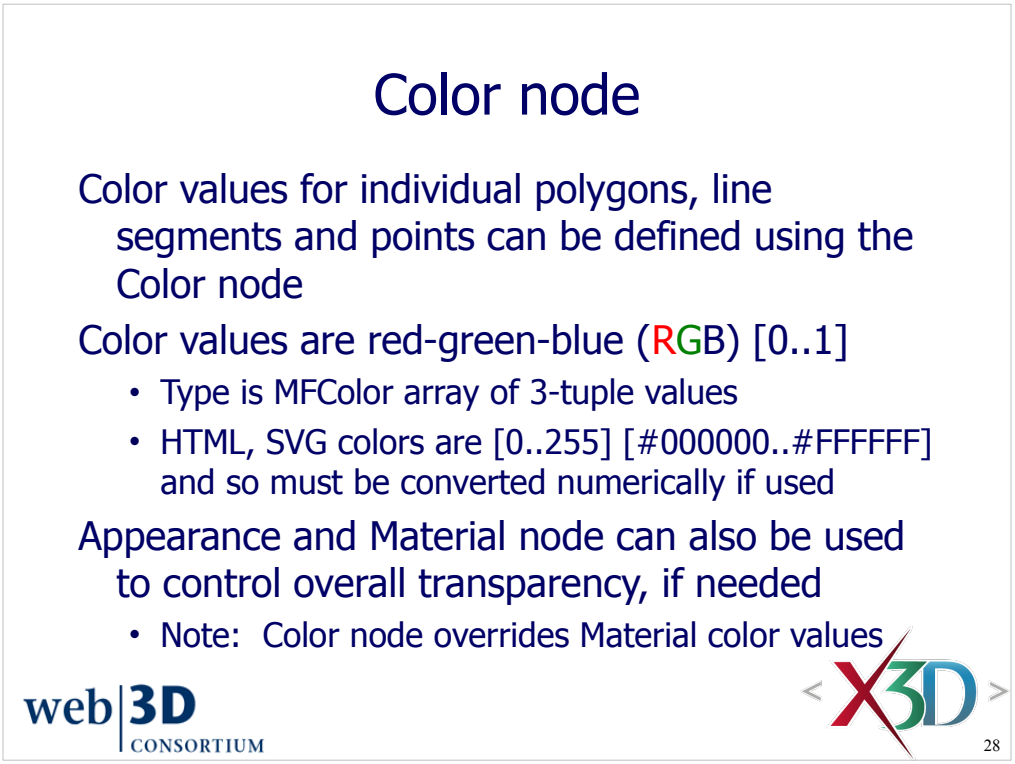
http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/Color.x3d

Coordinate	Coordinate builds geometry using a set of 3D coordinates. Coordinate is used by IndexedFaceSet, IndexedLineSet, LineSet and PointSet. Coordinate is also used by NurbsPositionInterpolator and NurbsOrientationInterpolator.
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!
point	[point: accessType inputOutput, type MFVec3f CDATA #IMPLIED] point contains a set of 3D coordinates.
containerField	[containerField: NMTOKEN "coord"] 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.

CoordinateDouble	CoordinateDouble builds geometry using a set of 3D coordinates. CoordinateDouble is used by IndexedFaceSet, IndexedLineSet, LineSet and PointSet. CoordinateDouble is also used by NurbsPositionInterpolator and NurbsOrientationInterpolator.
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!
point	[point: accessType inputOutput, type MFVec3d CDATA #IMPLIED] point contains a set of 3D coordinates.
containerField	[containerField: NMTOKEN "coord"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.
class	[class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.

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

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



Commas are allowed as whitespace characters, but the X3D Schema will return an XML validation error if a comma appears within a given SFColor 3-tuple value. This is stricter in the XML .x3d encoding than is otherwise seen in the ClassicVRML .x3dv encoding (which allows commas to go anywhere).

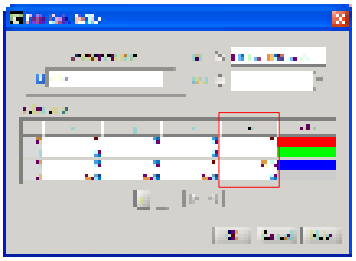
X3D Canonicalization (C14N) reformatting removes intermediate comma characters.

[#000000..#FFFFFF] are hexadecimal (base 16) values. These are often used in HTML for web pages.

Color and ColorRGBA nodes can appear as the color field within the following nodes: PointSet, IndexedLineSet, IndexedFaceSet, IndexedTriangleFanSet, IndexedTriangleSet, IndexedTriangleStripSet, TriangleFanSet, TriangleSet, and TriangleStripSet.

ColorRGBA node

- ColorRGBA used similarly to Color node, but adds alpha (opacity) to each red-green-blue value
  - alpha component equals (1 – transparency)
- Alpha values range 0 to 1
  - 0 means fully transparent
  - 1 means fully opaque
- RGBA values selectively allow transparent parts in geometry
  - Rather than single Material transparency consistently across full geometry with Color node



web3D CONSORTIUM

<X3D>

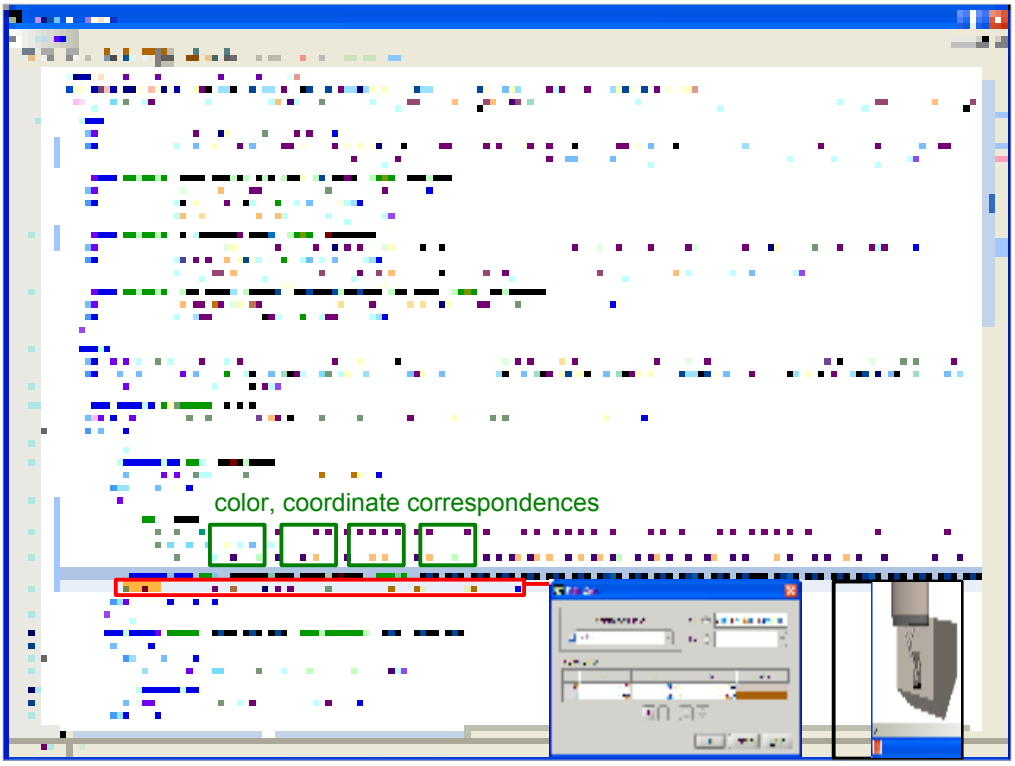
29

Note that, conceptually, alpha = (1 – transparency). There is no transparency field in Color/ColorRGBA nodes, though there is a transparency field in Material nodes.

Commas are allowed as whitespace characters, but the X3D Schema will return an XML validation error if a comma appears within a given SFColorRGBA 4-tuple value.

X3D Canonicalization (C14N) reformatting removes intermediate comma characters.

Color and ColorRGBA nodes can appear as the color field within the following nodes: PointSet, IndexedLineSet, IndexedFaceSet, IndexedTriangleFanSet, IndexedTriangleSet, IndexedTriangleStripSet, TriangleFanSet, TriangleSet, and TriangleStripSet.



X3D for Web Authors, Figure 6.3, p. 164

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/Color.x3d

Color	Color node defines a set of RGB color values. Color is only used by ElevationGrid, IndexedFaceSet, IndexedLineSet, LineSet and PointSet. Hint: colors are often controlled by Material instead.
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!
color	[color: accessType inputOutput, type MFColor CDATA #IMPLIED] color defines a set of RGB colors.
containerField	[containerField: NMTOKEN "color"] 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.

ColorRGBA	ColorRGBA node defines a set of RGBA color values. ColorRGBA is only used by ElevationGrid, IndexedFaceSet, IndexedLineSet, LineSet and PointSet. Hint: colors are often controlled by Material instead. Hint: alpha channel may be ignored under Interchange profile.
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!
color	[color: accessType inputOutput, type MFColorRGBA CDATA #IMPLIED] color defines a set of RGBA colors.
containerField	[containerField: NMTOKEN "color"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.
class	[class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.

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

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

PointSet node

PointSet creates a series of simple unconnected points in 3D space
- Contains Coordinate node for *point* data
- Since points are separate, *coordIndex* unnecessary

Each point typically drawn as a single pixel
- Or consistently as multiple pixels
- Thus scaling and perspective are quite deceiving
- Rarely used due to perspective inconsistencies

Color can be set in one of two ways
- Uniformly via Material *emissiveColor* value
- Individually via contained Color/ColorRGBA node

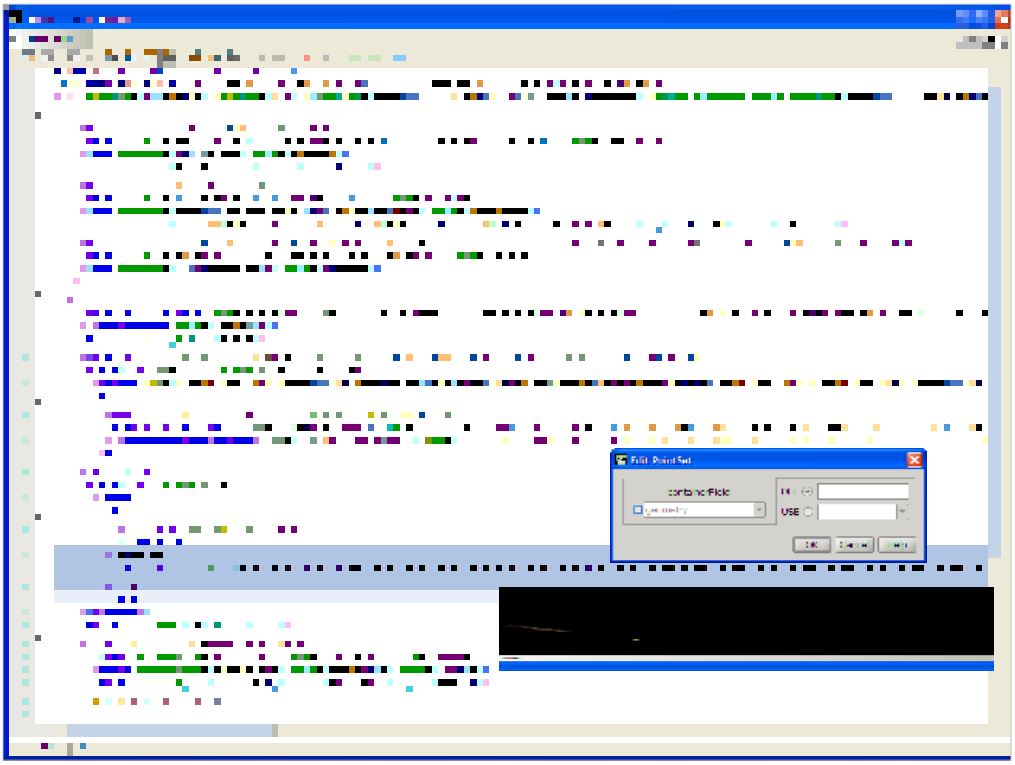
Note that Material *diffuseColor* and other fields have no effect on PointSet node.

The term *pixel* is an abbreviation for picture element, that is, the smallest drawable color on a display screen. Because pixel size can vary quite widely depending on screen resolution, authors thus have little control over the actual on-screen size of each point. Different displays and different resolutions can make points look quite different. Thus pixels are actually a hardware-dependent concept.

Points are not lit, are not texture-mapped, and do not participate in collision detection.

When properly constructed, point-map renderings can be quite interesting.

In recent years, there has been a lot of 3D graphics research progress in point-based rendering (PBR). Hopefully some of these capabilities will get introduced in future versions of X3D.



X3D for Web Authors, Figure 6.4, p. 167

Points are often hard to see, especially when projected since projectors typically do not have the same resolution quality or color fidelity as computer monitors.

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/PointSet.x3d

PointSet	**PointSet is a node that contains a set of colored 3D points, represented by contained Color and Coordinate nodes. Color values or a Material emissiveColor is used to draw lines and points.** **Hint: use a different color (or emissiveColor) than the background color.** **Hint: insert a Shape node before adding geometry or Appearance. You can also substitute a type-matched ProtoInstance for content.**
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!
containerField	**[containerField: NMTOKEN "geometry"]** containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.
class	**[class CDATA #IMPLIED]** class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.

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

IndexedLineSet node

IndexedLineSet creates an array of line segments

- Contains Coordinate node for *point* data
- Can be discontinuous or share points repeatedly
- Each set of connected line segments is a *polyline*

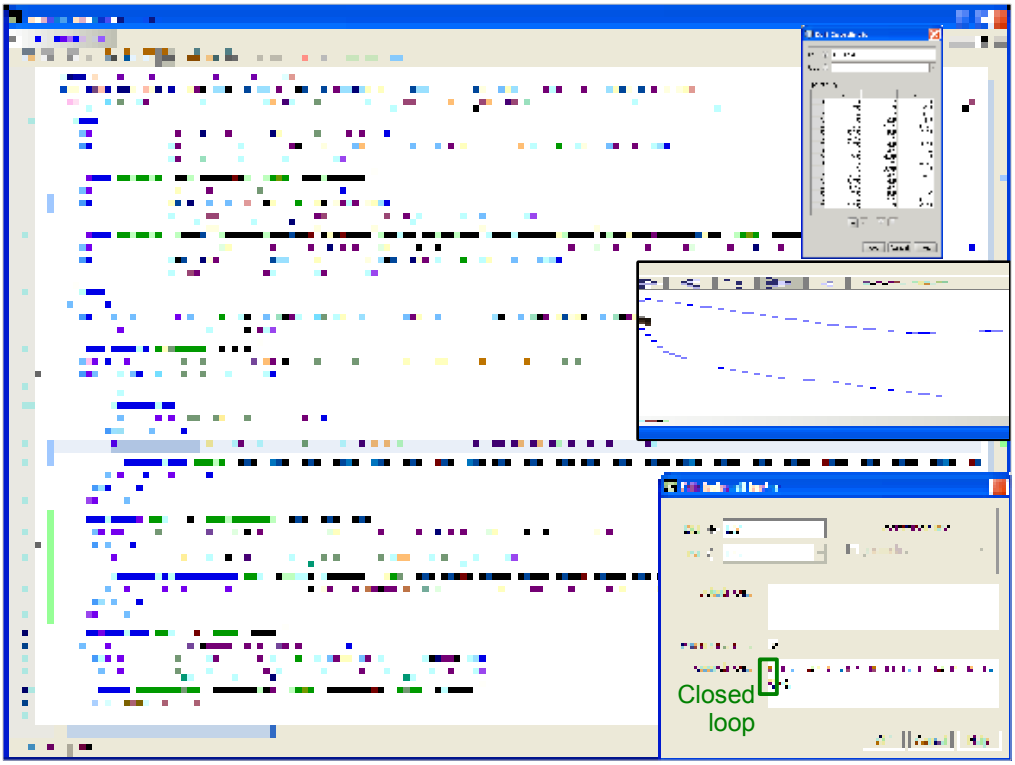
Lines are not lit, use no texture-mapped images, and do not participate in collision detection

Color can be set in one of two ways

- Uniformly via Material *emissiveColor* value — Not *diffuseColor*!
- Individually via contained Color/ColorRGBA node; applied either by individual points, or by each segment, as determined by *colorPerVertex*

Note that Material *diffuseColor* and other fields have no effect on IndexedLineSet.

This means that your line will be invisible (*emissiveColor* black) unless you modify your Material node!

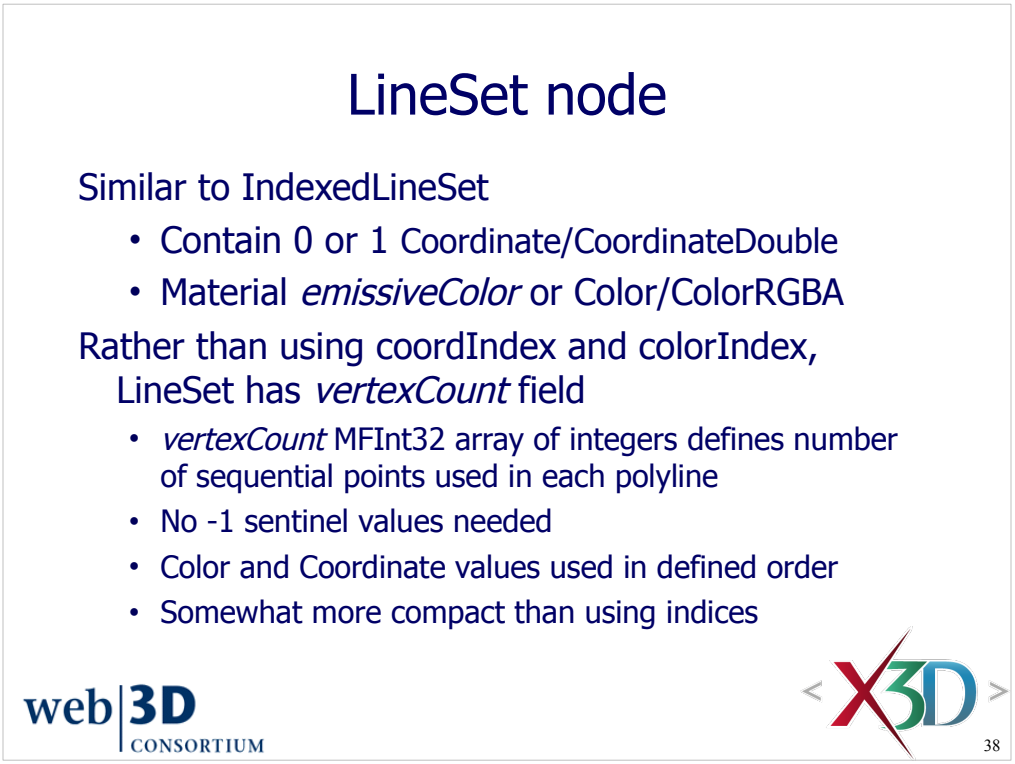


X3D for Web Authors, Figure 6.5, p. 170

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/IndexedLineSet.x3d

IndexedLineSet	**IndexedLineSet is a geometry node that can contain a Color node and a Coordinate node. Color values or a Material emissiveColor is used to draw lines and points. Lines are not lit, are not texture-mapped, and do not participate in collision detection.** **Hint: use a different color (or emissiveColor) than the background color.** **Hint: if rendering Coordinate points originally defined for an IndexedFaceSet, index values may need to repeat each initial vertex to close each polygon outline. Step-wise colors or linear interpolation of colors can be used as a good scientific visualization technique to map arbitrary function values to a color map.** **Hint: insert a Shape node before adding geometry or Appearance. You can also substitute a type-matched ProtoInstance for content.**
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!
coordIndex	**[coordIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** coordIndex indices provide order in which coordinates are applied. Order starts at index 0, commas are optional between sets, use -1 to separate indices for each polyline. **Hint:** if rendering Coordinate points originally defined for an IndexedFaceSet, index values may need to repeat initial each initial vertex to close the polygons.
colorPerVertex	**[colorPerVertex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether Color node is applied per vertex (true) or per polyline (false).
colorIndex	**[colorIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** colorIndex indices provide order in which colors are applied. **Hint:** if rendering Coordinate points originally defined for an IndexedFaceSet, index values may need to repeat initial each initial vertex to close the polygons.
set_coordIndex	**[set_coordIndex: accessType inputOnly, type MFInt32 CDATA #FIXED ""]** coordIndex indices provide order in which coordinates are applied. Order starts at index 0, commas are optional between sets. Use -1 to separate indices for each polygon.
set_colorIndex	**[set_colorIndex: accessType initializeOnly, type MFInt32 CDATA #FIXED ""]** colorIndex indices provide order in which colors are applied.
containerField	**[containerField: NMTOKEN "geometry"]** containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.
class	**[class CDATA #IMPLIED]** class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.

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

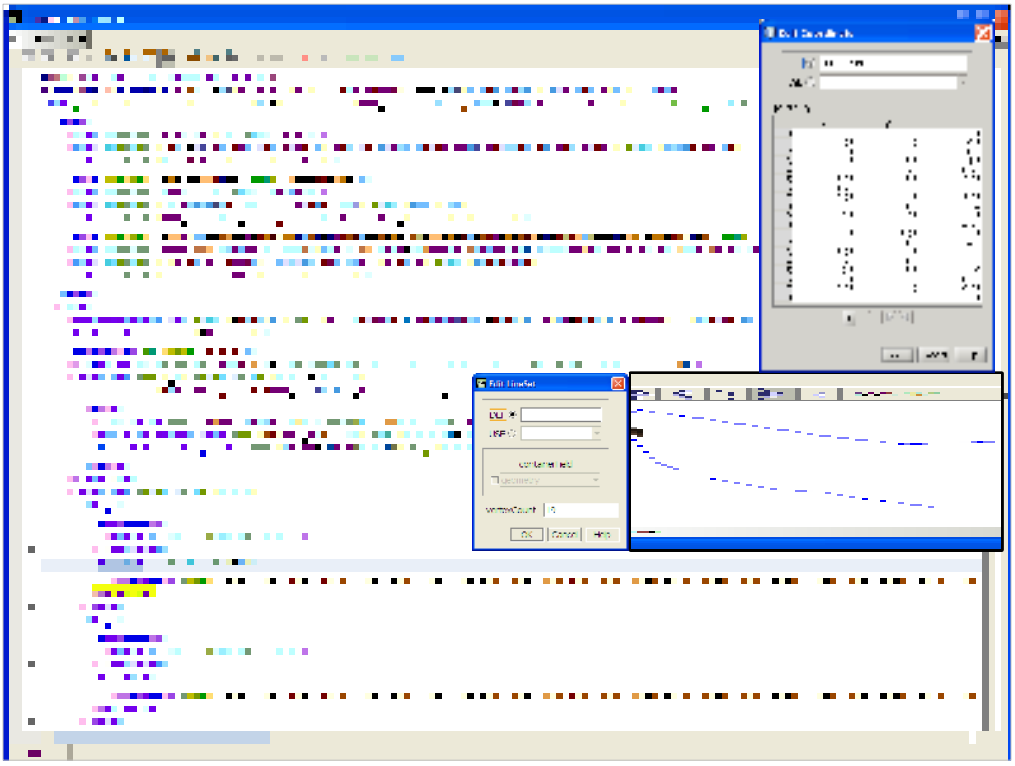


Note that Material *diffuseColor* and other fields have no effect on LineSet node.

LineSet was not part of VRML97 specification, it was introduced in X3D v3.0.

Things for you to do:

- Compare wireframe rendering mode
- Optical illusion when spinning line cube due to constant line width



X3D for Web Authors, Figure 6.5, p. 170

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/IndexedLineSet.x3d

Note that we had to duplicate the initial point (0 -7 -1) as the last value, so that this collection of 19 points produced a closed loop.

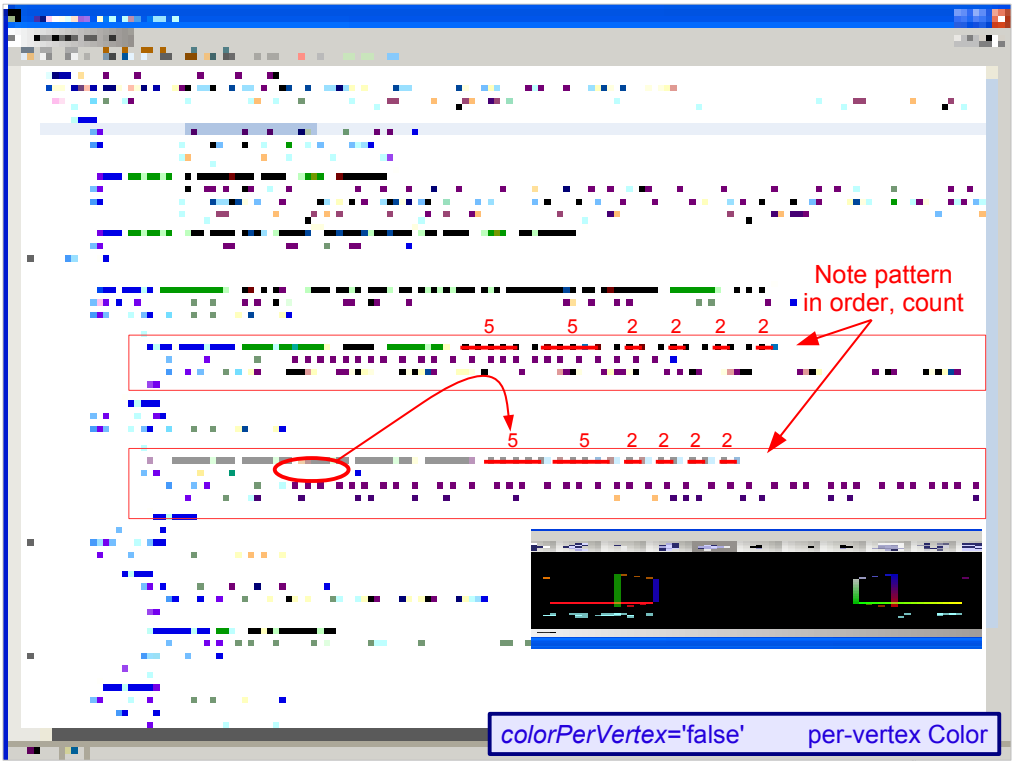
Notice the size tradeoff: sometimes one form is bigger, sometimes the other form is bigger.

- For IndexedLineSet we had 18 Coordinate values and 19 *coordIndex* values.
- For LineSet we had 19 Coordinate values and 1 *vertexCount* value.

So if you ask, “which should I use, IndexedLineSet of LineSet?” the answer is “It depends.”

- If you care about file size, check it yourself (by example or by calculation)
- If you care about ease of authoring, just use whichever you are comfortable with.

Appropriate acronym here is YMMV: Your Mileage May Vary. ☺ ☻



ILS = IndexedLineSet, LS=LineSet

IndexedLineSet example shows *colorPerVertex*='false' so that each polyline segment is a single color.

Note that LineSet can only be colored by vertex (and does not have a *colorPerVertex* field).

LineSet	**LineSet is a geometry node that can contain a Color node and a Coordinate node. Color values or a Material emissiveColor is used to draw lines and points. Lines are not lit, are not texture-mapped, and do not participate in collision detection.** **Hint: use a different color (or emissiveColor) than the background color. Linear interpolation of colors can be used as a good scientific visualization technique to map arbitrary function values to a color map.** **Hint: insert a Shape node before adding geometry or Appearance. You can also substitute a type-matched ProtoInstance for content.**
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!
vertexCount	**[vertexCount: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** [2,infinity) vertexCount describes how many vertices are used in each polyline from Coordinate field. Coordinates are assigned to each line by taking vertexCount[n] vertices from Coordinate field.
containerField	**[containerField: NMTOKEN "geometry"]** containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.
class	**[class CDATA #IMPLIED]** class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.

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

IndexedFaceSet node 1

IndexedFaceSet creates a set of polygons (faces)

- Contains Coordinate node for *point* data
- Can be discontinuous or share points repeatedly
- You can essentially create any geometry with IFS

Color can be set in one of two ways

- Uniformly via sibling Material fields
- Individually via contained Color/ColorRGBA node; applied either by individual points, or by each polygon, as determined by *colorPerVertex*

web3D CONSORTIUM

42

All Material fields are active, relevant for IndexedFaceSet and other polygonal nodes.

IndexedFaceSet node 2

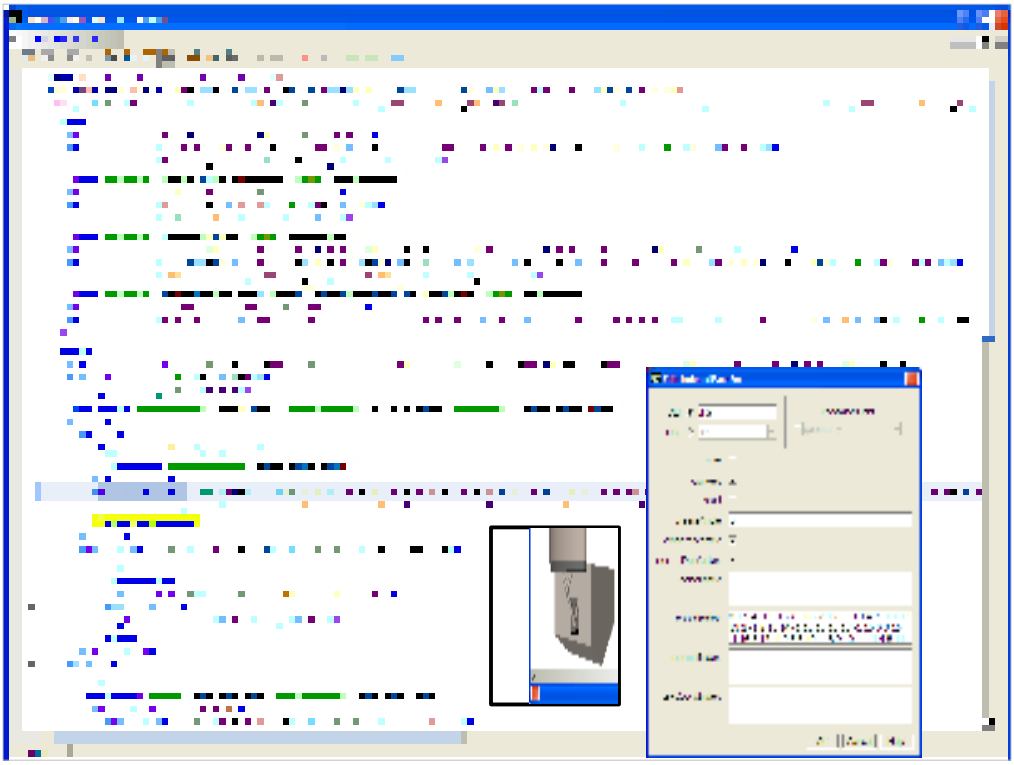
Many fields and features apply

- *ccw*, *convex*, *solid*, *creaseAngle* as before
- *colorPerVertex*, *normalPerVertex* as before
- *coordIndex, colorIndex, normalIndex* as before
- *texCoordIndex* applies texture coordinates to map texture images to individual geometry points

Contained nodes (0 or 1 of each)

- Coordinate/CoordinateDouble (essential, required)
- Color/ColorRGBA
- Normal, TextureCoordinate

web3D CONSORTIUM



X3D for Web Authors, Figure 6.7, p. 175

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/IndexedFaceSet.x3d

IndexedFaceSet	**IndexedFaceSet is a geometry node that can contain a Color, Coordinate, Normal and TextureCoordinate node.** **Hint: insert a Shape node before adding geometry or Appearance. You can also substitute a type-matched ProtoInstance for content.**
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!
coordIndex	**[coordIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** coordIndex indices provide order in which coordinates are applied. Order starts at index 0, commas are optional between sets. Use -1 to separate indices for each polygon.
ccw	**[ccw: accessType initializeOnly, type SFBool (true|false) "true"]** ccw = counterclockwise: ordering of vertex coordinates orientation. **Hint:** ccw false can reverse solid (backface culling) and normal-vector orientation.
convex	**[convex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether all polygons in a shape are convex (true), or possibly concave (false) A convex polygon is planar, does not intersect itself, and has all interior angles < 180 degrees. **Interchange profile hint:** only convex=true IndexedFaceSets are supported. **Warning:** concave geometry may be invisible default convex=true.
solid	**[solid: accessType initializeOnly, type SFBool (true|false) "true"]** Setting solid true means draw only one side of polygons (backface culling on), setting solid false means draw both sides of polygons (backface culling off). **Warning:** default value true can completely hide geometry if viewed from wrong side!
creaseAngle	**[creaseAngle: accessType initializeOnly, type SFFloat CDATA "0"]** [0..infinity) creaseAngle defines angle (in radians) for determining whether adjacent polygons are drawn with sharp edges or smooth shading. If angle between normals of two adjacent polygons is less than creaseAngle, smooth shading is rendered across the shared line segment. **Interchange profile hint:** only 0 and Î radians supported. **Hint:** creaseAngle=0 means render all edges sharply, creaseAngle=3.14 means render all edges smoothly.
colorPerVertex	**[colorPerVertex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether Color node is applied per vertex (true) or per polygon (false).
colorIndex	**[colorIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** colorIndex indices provide order in which colors are applied.
normalPerVertex	**[normalPerVertex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether Normal node is applied per vertex (true) or per polygon (false).
normalIndex	**[normalIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** **Interchange profile hint:** this field may be ignored.

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

texCoordIndex	**[texCoordIndex: accessType initializeOnly, type MFInt32 CDATA #IMPLIED]** List of texture-coordinate indices mapping attached texture to corresponding coordinates. **Hint:** use a tool!
set_coordIndex	**[set_coordIndex: accessType inputOnly, type MFInt32 CDATA #FIXED ""]** coordIndex indices provide order in which coordinates are applied. Order starts at index 0, commas are optional between sets. Use -1 to separate indices for each polygon.
set_colorIndex	**[set_colorIndex: accessType initializeOnly, type MFInt32 CDATA #FIXED ""]** colorIndex indices provide order in which colors are applied.
set_normalIndex	**[set_normalIndex: accessType inputOnly, type MFInt32 CDATA #FIXED ""]** **Interchange profile hint:** this field may be ignored.
set_texCoordIndex	**[set_texCoordIndex: accessType inputOnly, type MFInt32 CDATA #FIXED ""]** List of texture-coordinate indices mapping attached texture to corresponding coordinates. **Hint:** use a tool!
containerField	**[containerField: NMTOKEN "geometry"]** containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.
class	**[class CDATA #IMPLIED]** class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.

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

ElevationGrid node

ElevationGrid takes a rectangular array of floats and converts *height* array into post values above (or below) baseline y=0 ground plane

- *xDimension, zDimension* are row, column sizes
- *xSpacing*, *zSpacing* are lengths in meters
- *height* MFFloat array (size *xDimension* · *zDimension*)
- *ccw*, *solid* as before
- *colorPerVertex*, *normalPerVertex* as before

Contained nodes (0 or 1 of each)

- Color/ColorRGBA, Normal, TextureCoordinate

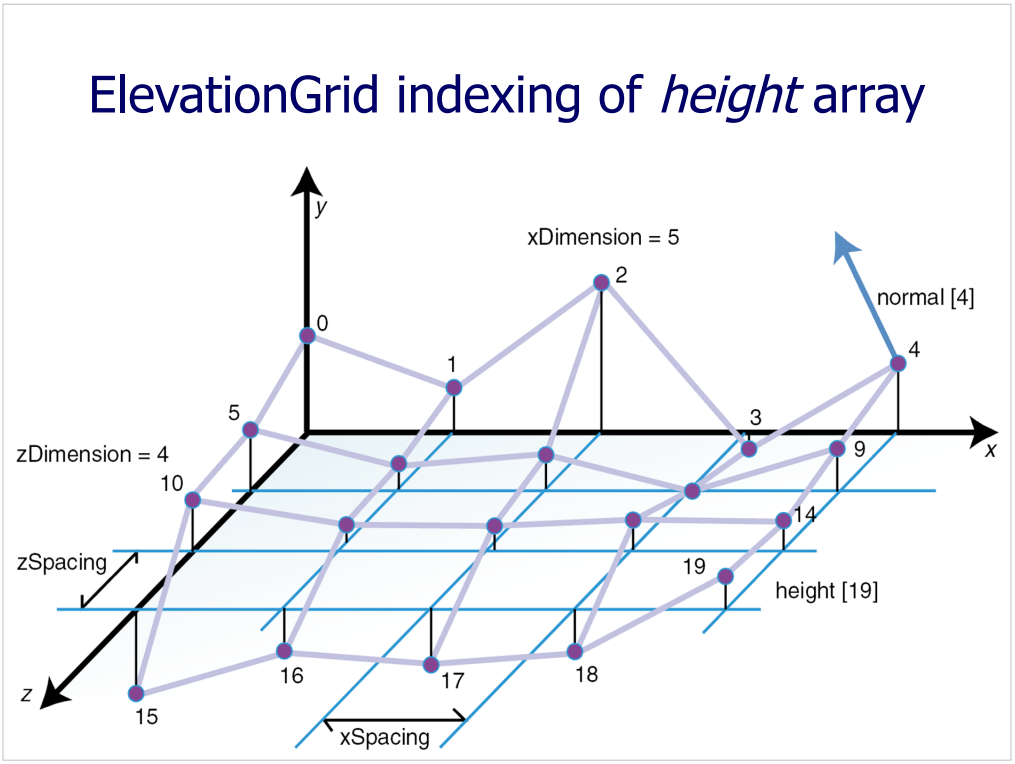
web3D CONSORTIUM

47

ElevationGrid is a good way to create terrain and some special shapes.

Related node: GeoElevationGrid in X3D Geospatial Component. A lot of work is going on that uses GeoElevationGrid as part of the X3D Earth project.

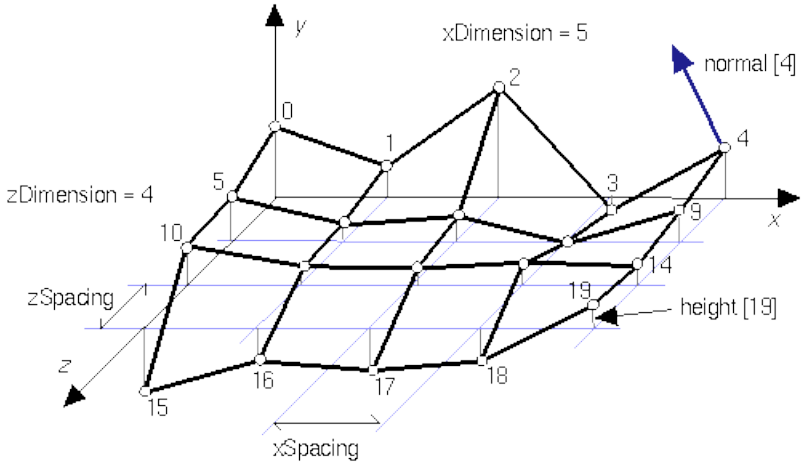
- http://www.web3d.org/x3d-earth
- https://x3d-earth.nps.edu



Indexing scheme for the ElevationGrid *height* array, including relationship to *xDimension*/*zDimension* and *xSpacing*/*zSpacing.*

X3D for Web Authors, Figure 6.8, p. 177

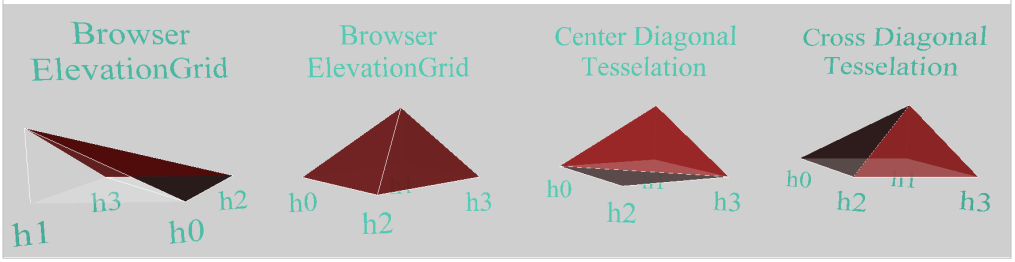
Adapted with permission from X3D Abstract Specification, section 13.3.4 ElevationGrid, Figure 13.4.



ElevationGrid inconsistencies due to noncoplanar quadrilaterals!

Alternate forms of tessellation are possible for nonplanar ElevationGrid quadrilaterals

- Almost all ElevationGrid quads are nonplanar, otherwise the geometry is flat
- Leftmost two figures show different views of grid
- Rightmost two figures show different tessellations
- Can avoid problem by using larger, fine-scaled grids



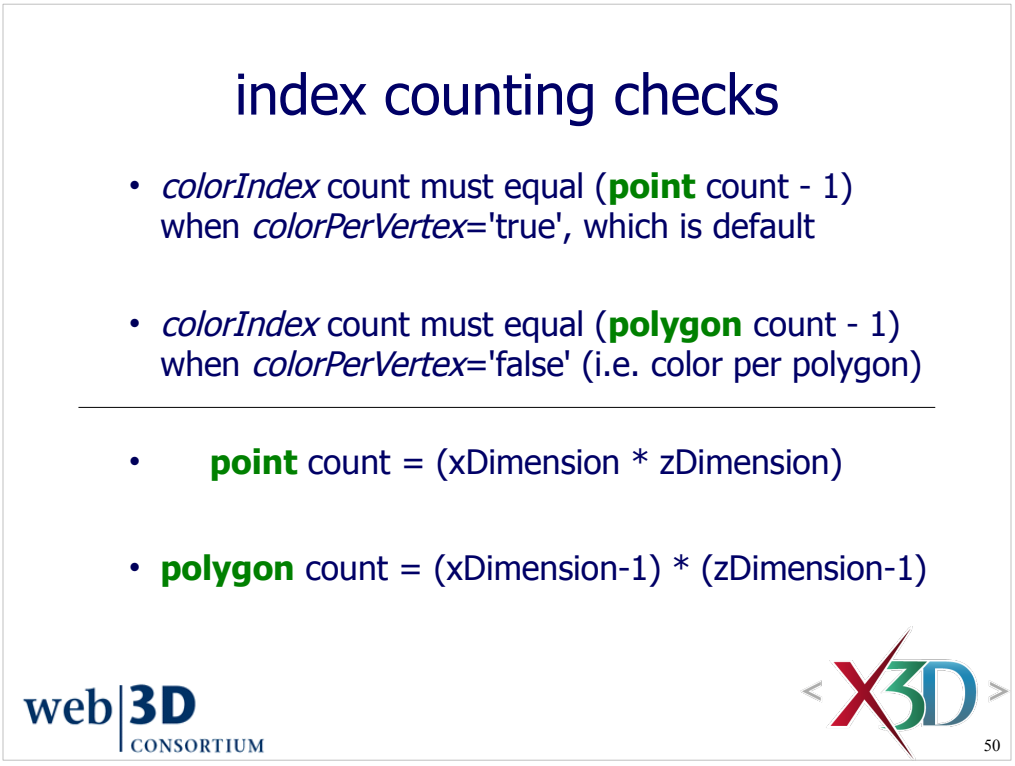
X3D for Web Authors, Figure 6.9, p. 178

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ElevationGridNonPlanarQuadrilaterals.x3d

Adding more and more elements to your ElevationGrid (i.e. bigger values for *xDimension* and *zDimension*) and a smooth value for *creaseAngle* won't eliminate this ambiguity in tesselating quadrilaterals into triangles, but it will make the differences so small that they aren't noticeable.

If you really really must control which way the triangles are split, then don't use ElevationGrid! Instead use IndexedFaceSet (or another triangular node) to strictly control each polygonal triangulation.

ElevationGrid can be very helpful for creating digital elevation terrain maps.



Failure to meet these requirements is an error. Results are often unpredictable since the X3D Specification doesn't provide strict requirements for how to handle errors.

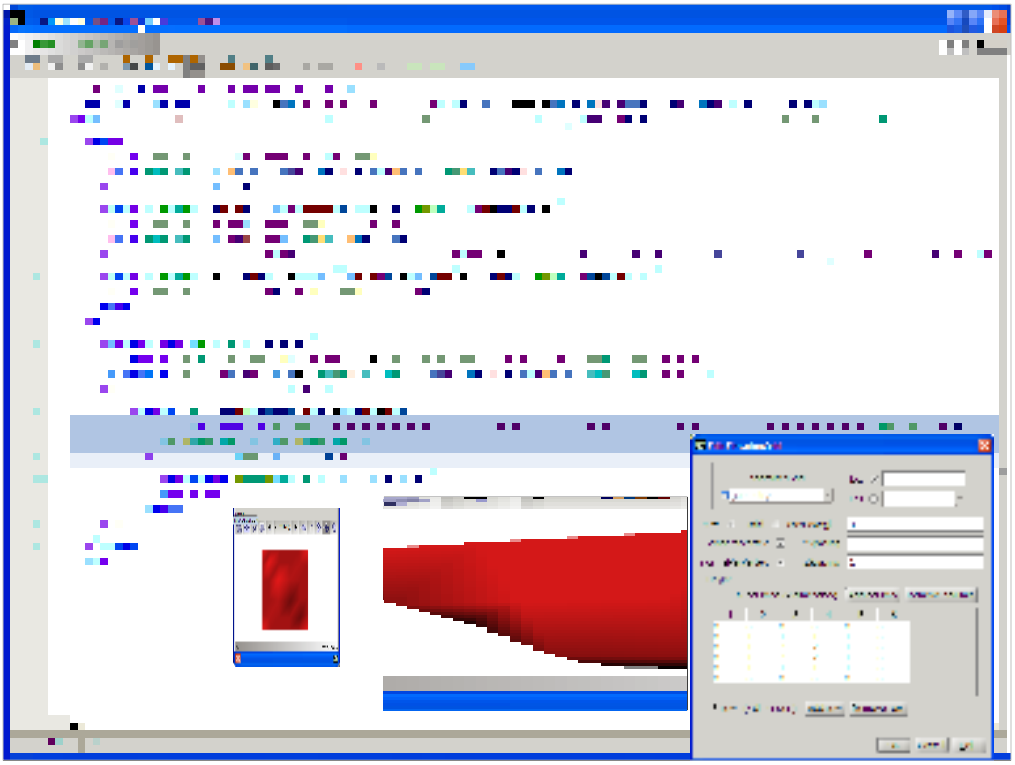
ElevationGrid node

Simple authoring trick

- Use a spreadsheet or some other simple tool to create a table of values, then cut/paste the values into the ElevationGrid *height* field
- Don't forget to also specify dimensions and spacing

Interesting authoring trick

- ElevationGrid does not have to lay flat on the horizontal plane, you can rotate it to another angle
- Example: stone canyon walls inside Kelp Forest exhibit



ElevationGrid	ElevationGrid is a geometry node. ElevationGrid is a rectangular grid of varying height above a flat surface. ElevationGrid can contain Color, Normal and TextureCoordinate nodes. Hint: insert a Shape node before adding geometry or Appearance. You can also substitute a type-matched ProtoInstance for content.
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!
xDimension	[xDimension: accessType initializeOnly, type SFInt32 CDATA "0"] Number of grid-array elements along X direction.
zDimension	[zDimension: accessType initializeOnly, type SFInt32 CDATA "0"] Number of grid-array elements along Z direction.
xSpacing	[xSpacing: accessType initializeOnly, type SFFloat CDATA "1.0"] Meters distance between grid-array vertices along X direction. Hint: total horizontal x-axis distance equals (xDimension-1) * xSpacing.
zSpacing	[zSpacing: accessType initializeOnly, type SFFloat CDATA "1.0"] Meters distance between grid-array vertices along Z direction. Hint: total vertical z-axis distance equals (zDimension-1) * zSpacing.
height	[height: accessType initializeOnly, type MFFloat CDATA #IMPLIED] Grid array of height vertices along upward Y direction, with xDimension rows and zDimension columns.
set_height	[set_height: accessType inputOnly, type MFFloat CDATA #FIXED ""] Grid array of height vertices along upward Y direction, with xDimension rows and zDimension columns.
ccw	[ccw: accessType initializeOnly, type SFBool (true|false) "true"] ccw = counterclockwise: ordering of vertex coordinates orientation. Hint: ccw false can reverse solid (backface culling) and normal-vector orientation.
creaseAngle	[creaseAngle: accessType initializeOnly, type SFFloat CDATA "0"] [0..infinity) creaseAngle defines angle (in radians) for determining whether adjacent polygons are drawn with sharp edges or smooth shading. If angle between normals of two adjacent polygons is less than creaseAngle, smooth shading is rendered across the shared line segment. Hint: creaseAngle=0 means render all edges sharply, creaseAngle=3.14 means render all edges smoothly.
solid	[solid: accessType initializeOnly, type SFBool (true|false) "true"] Setting solid true means draw only one side of polygons (backface culling on), setting solid false means draw both sides of polygons (backface culling off). Warning: default value true can completely hide geometry if viewed from wrong side!
colorPerVertex	[colorPerVertex: accessType initializeOnly, type SFBool (true|false) "true"] Whether Color node is applied per vertex (true) or per quadrilateral (false).
normalPerVertex	[normalPerVertex: accessType initializeOnly, type SFBool (true|false) "true"] Whether Normal node is applied per vertex (true) or per quadrilateral (false).
containerField	[containerField: NMTOKEN "geometry"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.
class	[class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.

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

Extrusion node 1

Extrusion begins with a planar *crossSection* outline, then stretches (extrudes) it along a *spine* polyline

- *crossSection* is MFVec2f array of 2-tuple floating-point 2D coordinate pairs creating an outline
- *spine* is MFVec3f array of 3-tuple floating-point 3D coordinates creating a polyline

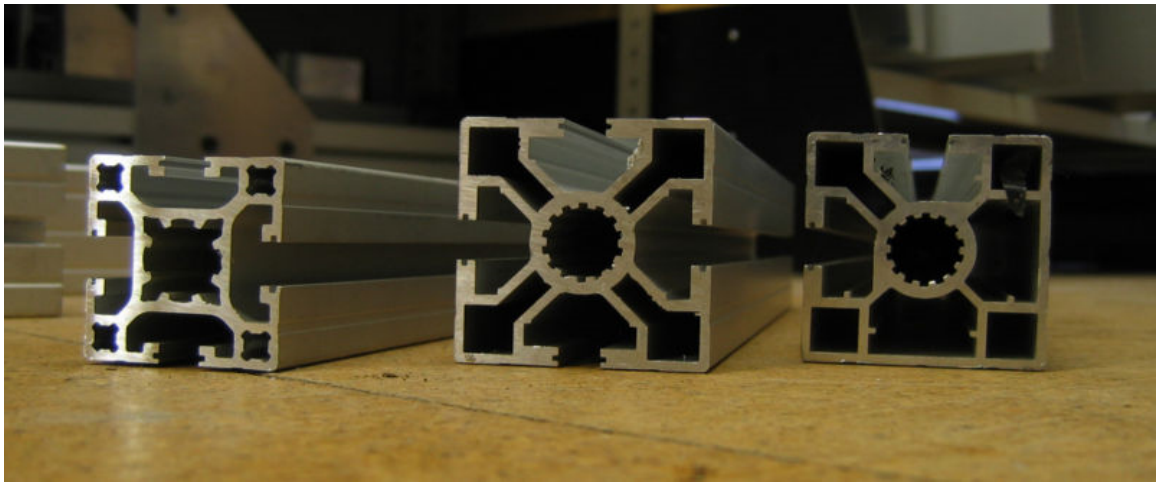
Extrusion is a bit tricky to master, but provides a great way to create sophisticated geometry with little effort

web|3D CONSORTIUM

http://commons.wikimedia.org/wiki/Image:Extruded_aluminium_section_x3.jpg

Three pieces of extruded aluminum. They are intended to be bolted together, using special connectors which fit into the ends and/or side groves, allowing for quick and neat construction of metal structures.

Used with permission.



Play-doh Fun Factory!

Here is an example real-world Extrusion

Thanks to Jeff Weekley for the Fun Factory! Good fun.

http://www.hasbro.com/playdoh/default.cfm?page=products&product_id=8994

http://www.youtube.com/watch?v=bptbVF9XETo

Play-Doh Fun Factory

- Squeeze, shape, mold and extrude all kinds of crazy shapes with this classic set!
- 2 shapemaking strips have 10 patterns for extruding long strips of PLAY-DOH compound!
- Includes two 5-oz. cans of modeling compound, extruder with 3 half molds, 2 shapemaking strips (10 designs) and trimmer knife.

ge=pro

Extrusion node 2

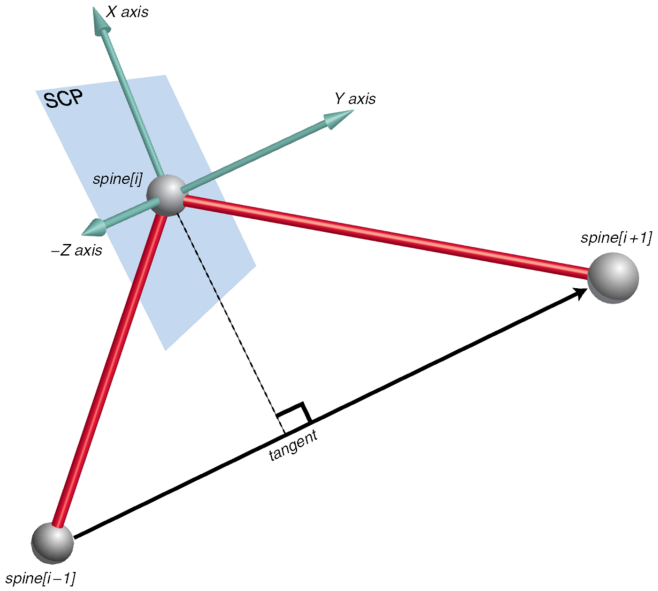
Spine-aligned cross-section plane (SCP) refers to each individual copy of the cross section, each of which appear about each *spine* point

- Extrusion outer hull simply connects corresponding points on these cross-section outlines
- If the outline of the Extrusion is degenerate or ill defined, then the polygons making up Extrusion outline are similarly confounded

Drawing simple outlines of *crossSection* on graph paper is great way to keep things straight

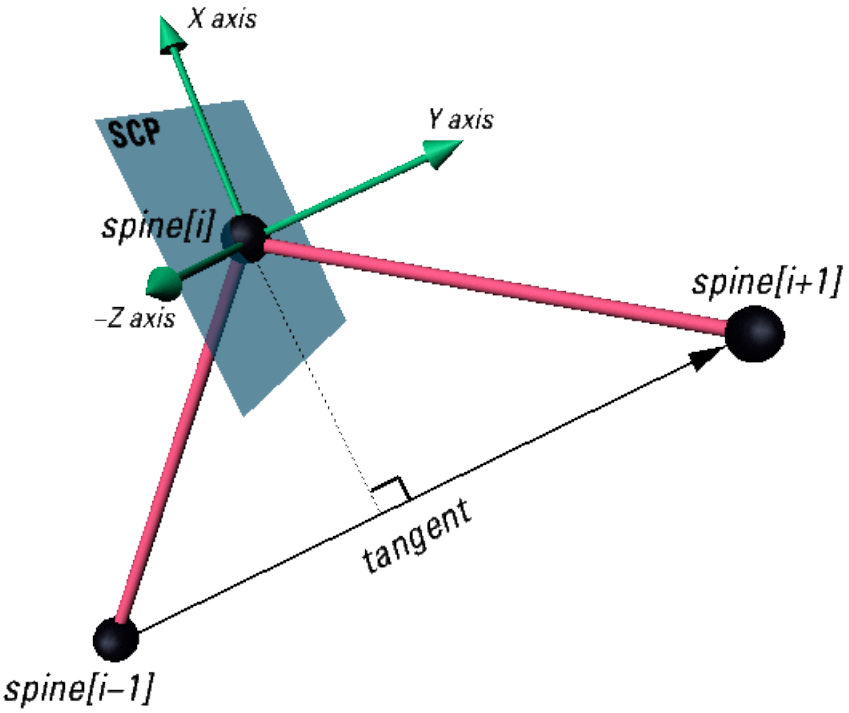
web3D CONSORTIUM

Extrusion spine-aligned cross-section plane (SCP)



X3D for Web Authors, Figure 6.10, p. 181

Adapted with permission from X3D Abstract Specification, section 13.3.5 Extrusion, Figure 13.5, Spine-aligned cross-section plane at a spine point.



Extrusion node 2

Further modifications include *scale*, *orientation* to modify each cross-section about SCP center

- *scale* is MFVec2f array of 2-tuple floating-point pairs to scale the local spine-aligned crossSection plane
- *orientation* is MFRotation array to rotate
- Single value affects all simultaneously, array affects each repeated cross-section individually

Other fields are common

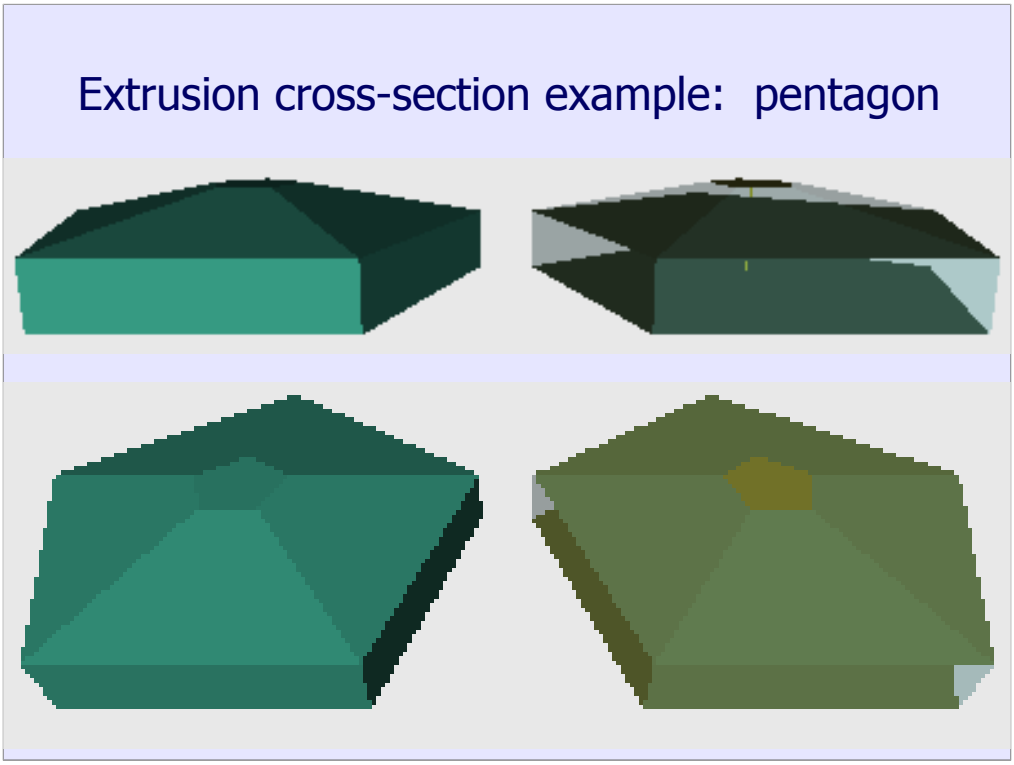
- *ccw*, *convex*, *solid*, *creaseAngle* as before
- *beginCap, endCap* are SFBool values to close ends

web3D CONSORTIUM

58

scale is easy to modify and get good shape changes at each SCP outline. Since it is a 2-tuple MFVec2f array, each scaling pair can be non-uniform.

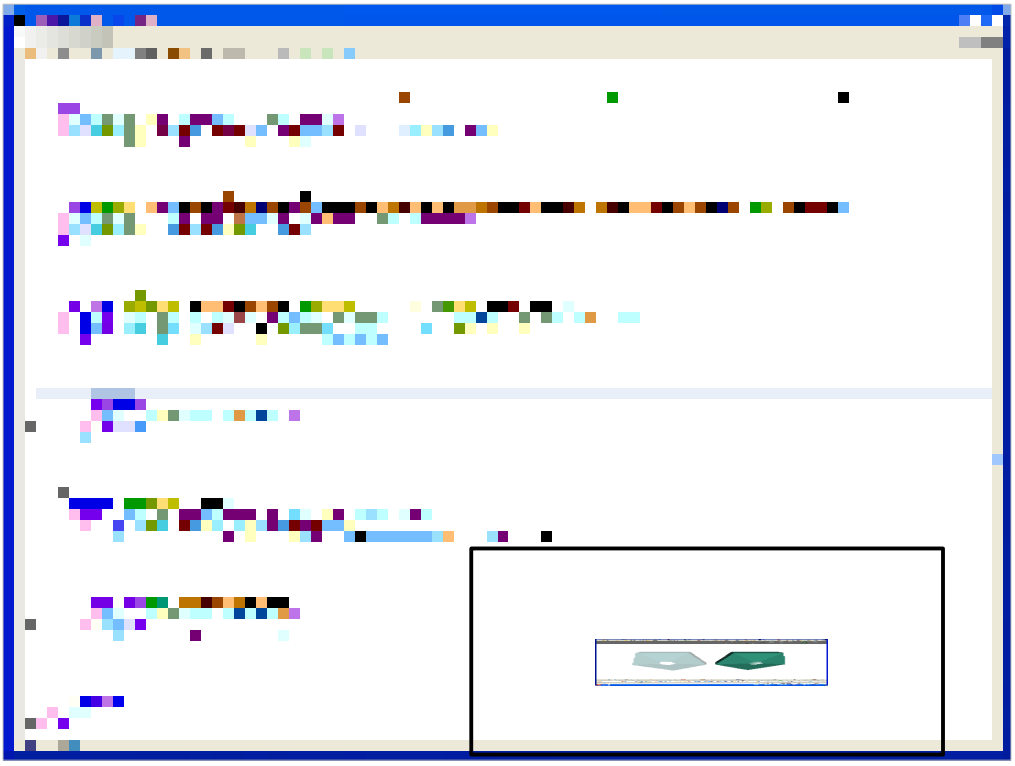
orientation is quite tricky, only attempt small rotation changes or significant errors can occur (such as geometry folding back on top of itself). Rotations are not cumulative but rather are applied at each of the SCP silhouettes, as indicated in the figure.



Example pentagon Extrusion views showing default rendering on left, and cross sections on right, with yellow *spine* line segment visible in upper right.

X3D for Web Authors, Figure 6.11, p. 182

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ExtrusionPentagon.x3d



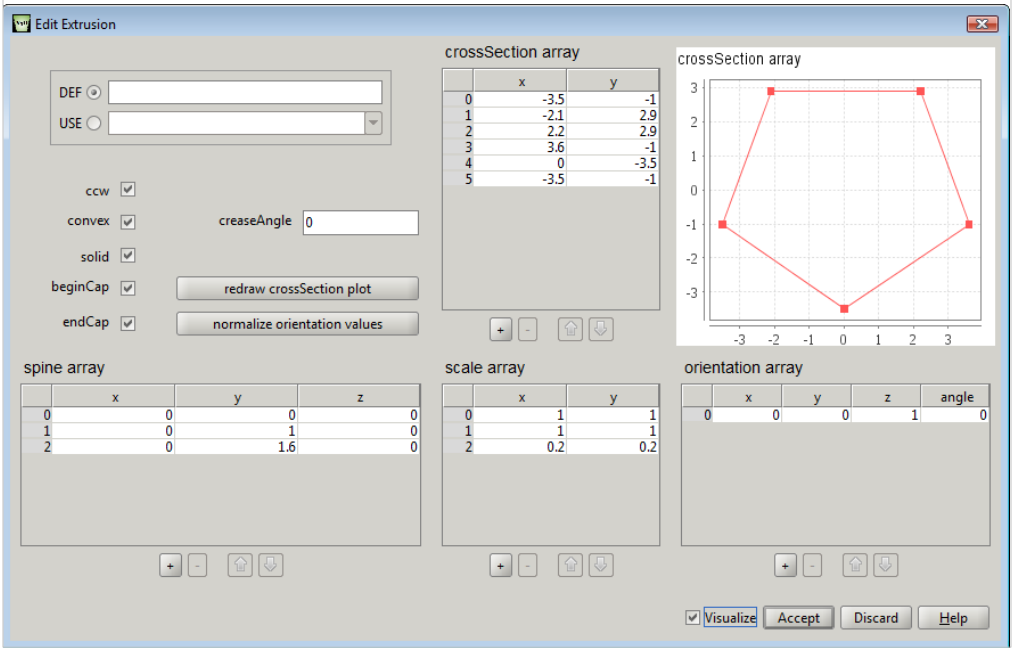
Example pentagon Extrusion views showing default rendering on left, and cross sections on right.

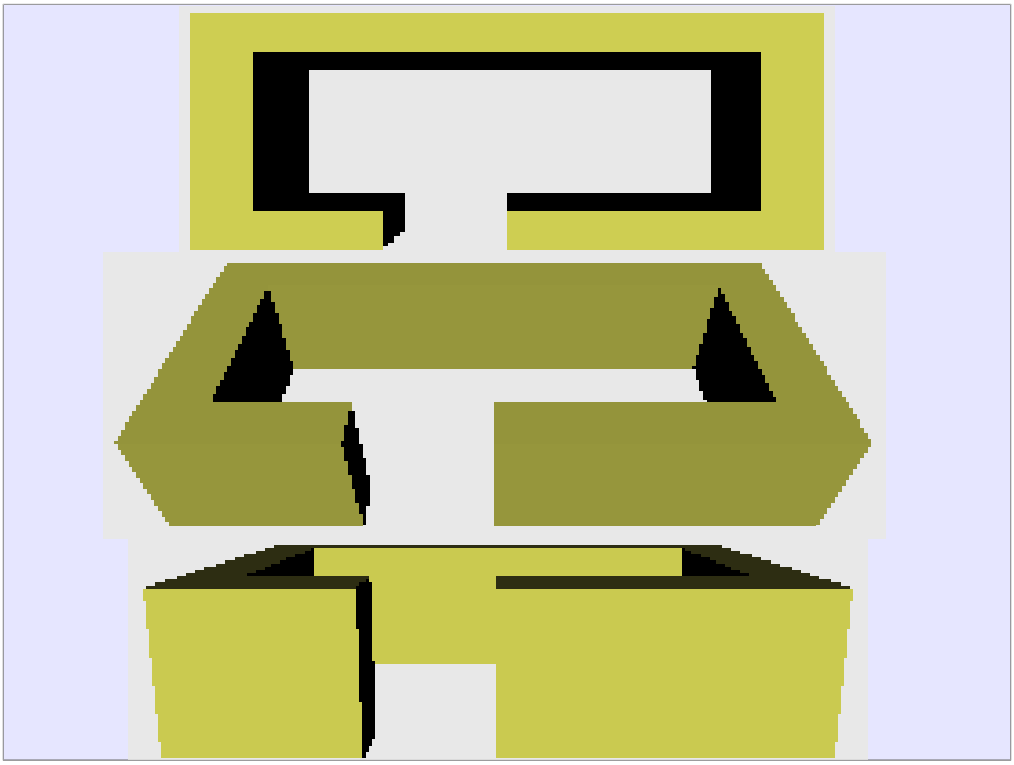
TODO: describe how to copy, adapt prototype ExternProtoDeclare and ProtoInstance

X3D for Web Authors, Figure 6.11, p. 182

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ExtrusionPentagon.x3d

X3D-Edit user interface for Extrusion

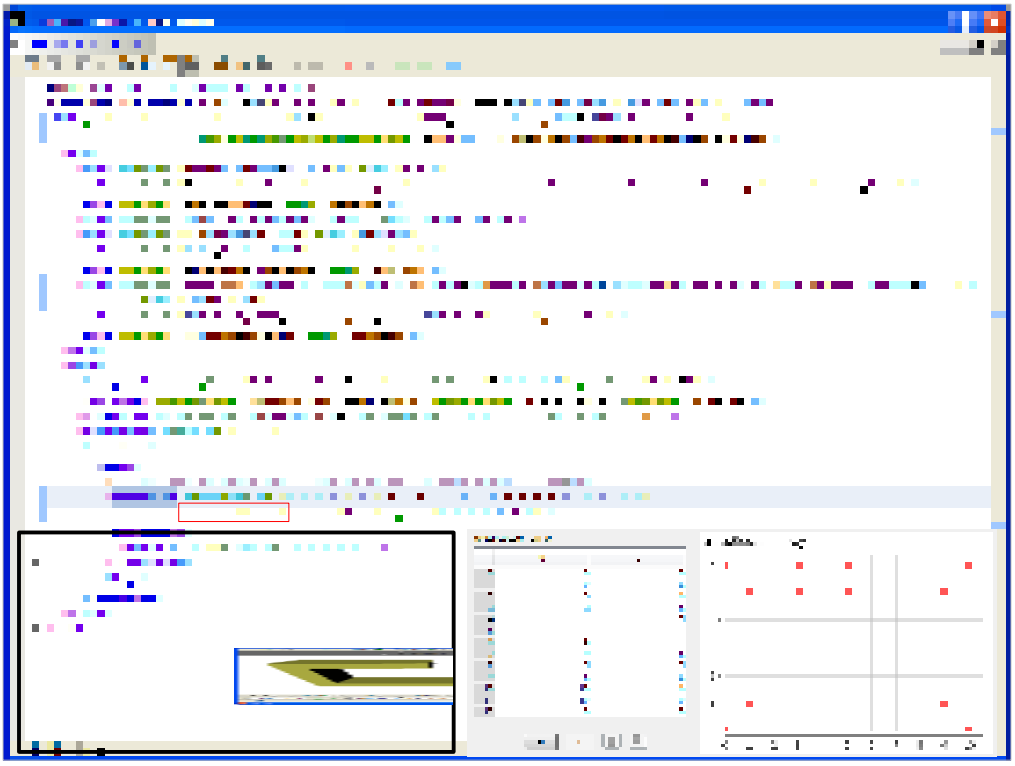




Extrusion example constructing the walls of a building.

X3D for Web Authors, Figure 6.12, p. 183

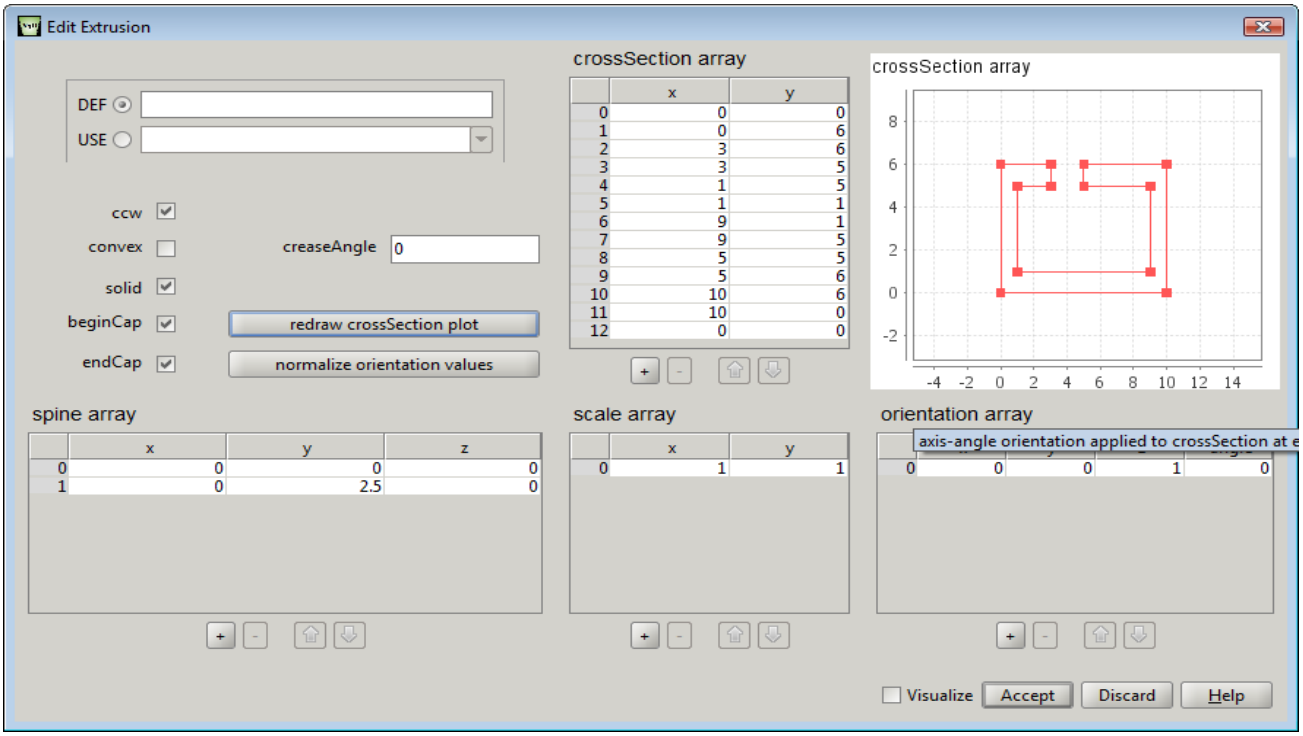
http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ExtrusionRoomWalls.x3d



Extrusion example constructing the walls of a building.

X3D for Web Authors, Figure 6.12, p. 183

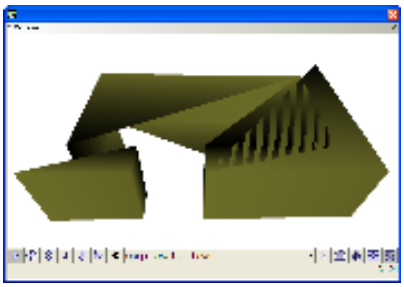
http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ExtrusionRoomWalls.x3d



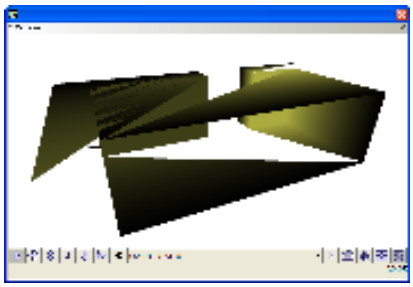
Concave geometry defects

Concave geometry with *convex*='true' can lead to confused geometry results

- Nonsensical polygons
- Aliasing (tearing) of coplanar polygons
- To correct: set *convex*='false'



Erroneous rendering



Erroneous rendering

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ExtrusionRoomWalls.x3d

Debugging Extrusion problems 1

Unlike most other nodes, ill-defined geometry is possible with Extrusion. Things to check:

- Verify proper *crossSection*, *spine*, *scale*, *orientation* array values and lengths
- Set *convex*='false' if geometry might be concave
- Set *solid*='false' to render inside and outside, eliminating "invisible geometry" when viewed from behind or inside the exterior hull
- Set *ccw*='false' if *crossSection* might be defined in clockwise direction

web3D CONSORTIUM

65

Debugging Extrusion problems 2

Counting checks

- Length of 2-tuple *scale* array must be 0, 1, or match length of 3-tuple *spine* array
- Length of 4-tuple *orientation* array must be 0, 1, or match length of 3-tuple *spine* array
- Values in *scale* and *crossSection* arrays must be multiple of 2 (MFVec2f)
- Values in *spine* array must be multiple of 3 (MFVec3f)
- Values in *orientation* array must be multiple of 4 (MFRotation)

web|3D CONSORTIUM

66

ExtrusionCrossSection prototype

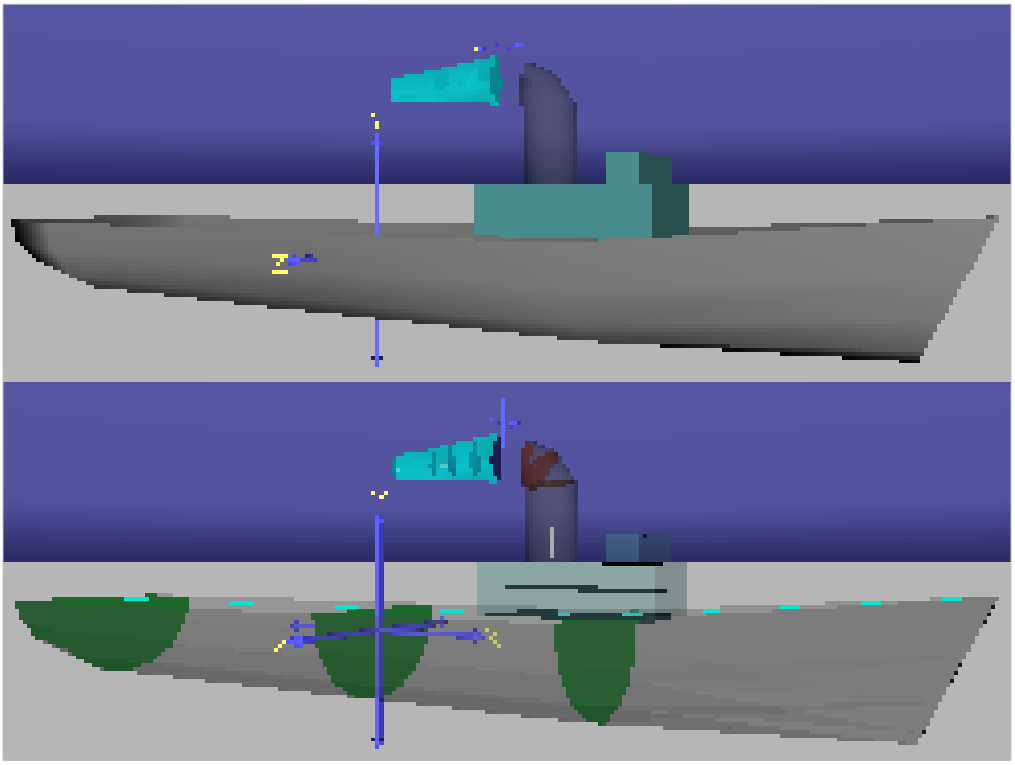
Preceding ExtrusionPentagon.x3d example scene contains a new construct: a Prototype node

- ExternProtoDeclare refers to external prototype url and defines field signatures
- ProtoInstance creates an instance of the new node
- fieldValue definitions provide parameter values, in this case the same values as Extrusion of interest

Result is a specially computed Extrusion showing *crossSection* planes, *spine*, transparent sides

- Can provide helpful insight and debugging support

web3D CONSORTIUM



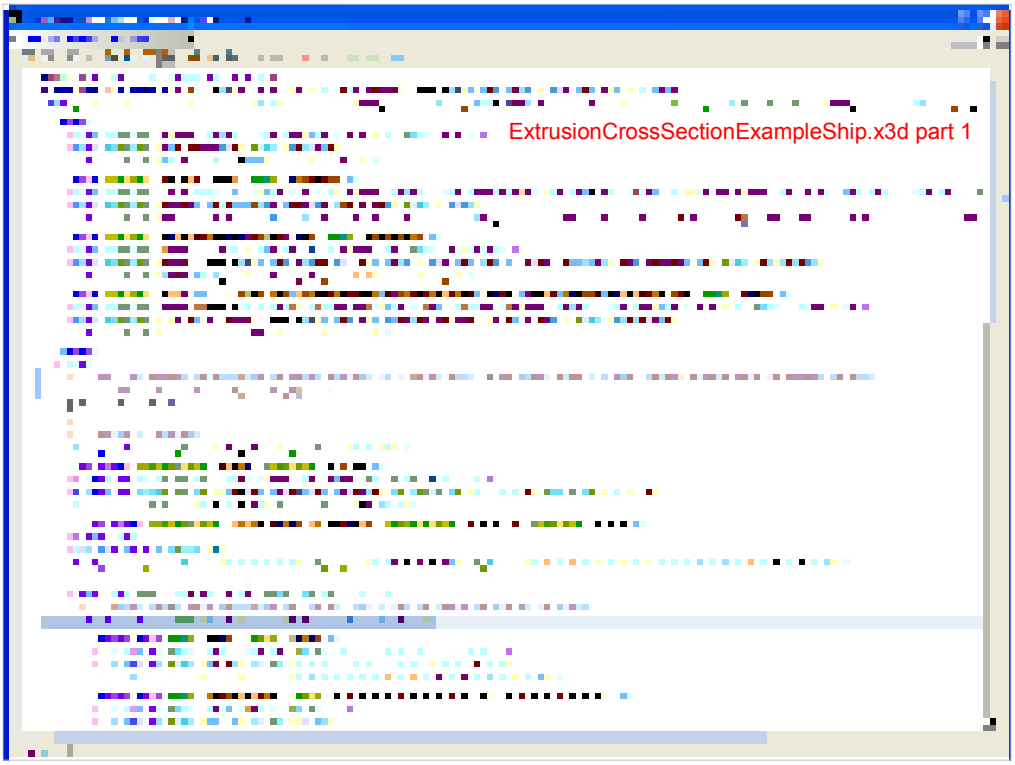
Extrusion example constructing the hull, superstructure, and smoke trail of a ship.

X3D for Web Authors, Figure 6.13, p. 184

http://www.web3d.org/x3d/content/examples/Basic/course/ExtrusionCrossSectionExampleShip.x3d

http://www.web3d.org/x3d/content/examples/Basic/course/ExtrusionCrossSectionPrototype.x3d

Chapter 14, Creating Prototype Nodes covers ProtoInstance and ExternProtoDeclare.



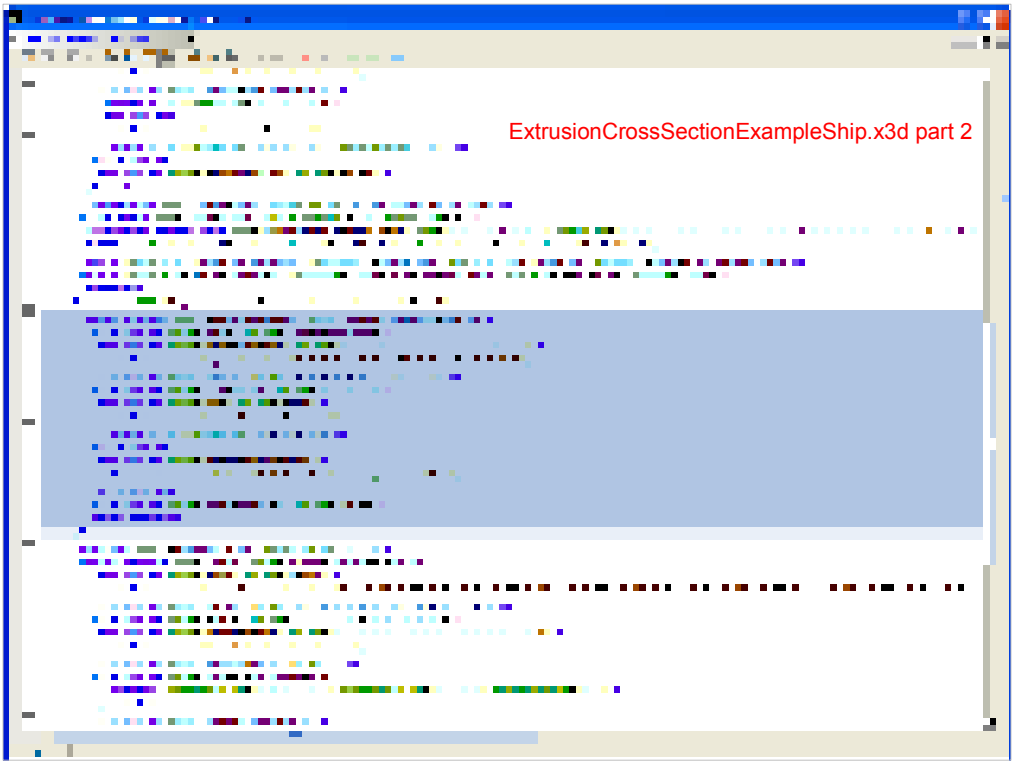
Extrusion example constructing the hull, superstructure, and smoke trail of a ship.

X3D for Web Authors, Figure 6.13, p. 184

http://www.web3d.org/x3d/content/examples/Basic/course/ExtrusionCrossSectionExampleShip.x3d

http://www.web3d.org/x3d/content/examples/Basic/course/ExtrusionCrossSectionPrototype.x3d

Chapter 14, Creating Prototype Nodes covers ProtoInstance and ExternProtoDeclare.



Extrusion example constructing the hull, superstructure, and smoke trail of a ship.

X3D for Web Authors, Figure 6.13, p. 184

http://www.web3d.org/x3d/content/examples/Basic/course/ExtrusionCrossSectionExampleShip.x3d

http://www.web3d.org/x3d/content/examples/Basic/course/ExtrusionCrossSectionPrototype.x3d

Chapter 14, Creating Prototype Nodes covers ProtoInstance and ExternProtoDeclare.

Extrusion	**Extrusion is a geometry node stretching a 2D cross section along a 3D-spine path in the local coordinate system Scaling/rotating cross-sections can produce a variety of shapes.** **Hint: insert a Shape node before adding geometry or Appearance.**
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!
spine	**[spine: accessType initializeOnly, type MFVec3f CDATA "0 0 0, 0 1 0"]** spine is a list of 3D points for a piecewise-linear curve forming a series of connected vertices, open or closed. This is the path along which the crossSection is extruded. **Hint:** number of spine points, scale values and orientation values must be the same.
crossSection	**[crossSection: accessType initializeOnly, type MFVec2f CDATA "1 1, 1 -1, -1 -1, -1 1, 1 1"]** An ordered set of 2D points drawing a piecewise-linear curve and forming a planar series of connected vertices. This provides a silhouette of the outer surface. **Warning:** match clockwise/counterclockwise or impossible/inverted geometry can result!
scale	**[scale: accessType initializeOnly, type MFVec2f CDATA "1 1"]** (0..infinity) scale is a list of 2D-scale parameters applied at each spine-aligned cross-section plane. **Hint:** number of spine points, scale values and orientation values must be the same. **Warning:** zero or negative scale values not allowed.
orientation	**[orientation: accessType initializeOnly, type MFRotation CDATA "0 0 1 0"]** orientation is a list of axis-angle orientation 4-tuples applied at each spine-aligned cross-section plane. **Hint:** number of spine points, scale values and orientation values must be the same.
beginCap	**[beginCap: accessType initializeOnly, type SFBool (true|false) "true"]** Whether beginning cap is drawn (similar to Cylinder top cap). **Warning:** cannot be changed after initial creation.
endCap	**[endCap: accessType initializeOnly, type SFBool (true|false) "true"]** Whether end cap is drawn (similar to Cylinder end cap). **Warning:** cannot be changed after initial creation.
ccw	**[ccw: accessType initializeOnly, type SFBool (true|false) "true"]** ccw = counterclockwise: ordering of vertex-coordinates orientation. **Hint:** ccw false can reverse solid (backface culling) and normal-vector orientation.
convex	**[convex: accessType initializeOnly, type SFBool (true|false) "true"]** Whether all polygons in a shape are convex (true), or possibly concave (false). A convex polygon is planar, does not intersect itself, and has all interior angles < 180 degrees. **Warning:** concave geometry may be invisible default convex=true.

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

creaseAngle	**[creaseAngle: accessType initializeOnly, type SFFloat CDATA "0.0"]** [0..infinity) creaseAngle defines angle (in radians) where adjacent polygons are drawn with sharp edges or smooth shading. If angle between normals of two adjacent polygons is less than creaseAngle, smooth shading is rendered across the shared line segment. **Hint:** creaseAngle=0 means render all edges sharply, creaseAngle=3.14 means render all edges smoothly.
solid	**[solid: accessType initializeOnly, type SFBool (true|false) "true"]** Setting solid true means draw only one side of polygons (backface culling on), setting solid false means draw both sides of polygons (backface culling off). **Warning:** default value true can completely hide geometry if viewed from wrong side!
set_crossSection	**[set_crossSection: accessType inputOnly, type MFVec2f CDATA #FIXED ""]** An ordered set of 2D points drawing a piecewise-linear curve and forming a planar series of connected vertices. This provides a silhouette of the outer surface. **Warning:** match clockwise/counterclockwise or impossible/inverted geometry can result!
set_orientation	**[set_orientation: accessType inputOnly, type MFRotation CDATA #FIXED ""]** orientation is a list of axis-angle orientation 4-tuples applied at each spine-aligned cross-section plane. **Hint:** number of spine points, scale values and orientation values must be the same.
set_scale	**[set_scale: accessType inputOnly, type MFVec2f CDATA #FIXED ""]** (0..infinity) scale is a list of 2D-scale parameters applied at each spine-aligned cross-section plane. **Hint:** number of spine points, scale values and orientation values must be the same. **Warning:** zero or negative scale values not allowed.
set_spine	**[set_spine: accessType inputOnly, type MFVec3f CDATA #FIXED ""]** spine is a list of 3D points for a piecewise-linear curve forming a series of connected vertices, open or closed. This is the path along which the crossSection is extruded. **Hint:** number of spine points, scale values and orientation values must be the same.
containerField	**[containerField: NMTOKEN "geometry"]** containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.
class	**[class CDATA #IMPLIED]** class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.

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

back to Table of Contents

Additional Resources

Geometry nodes

Chapter 2, Primitives

- Box, Cone, Cylinder, Sphere, Text / FontStyle

Chapter 6, Points Lines and Polygons

- PointSet, IndexedLineSet, IndexedFaceSet, ElevationGrid, Extrusion

We are here

Chapter 10, Geometry2D

- Arc2D,ArcClose2D, Circle2D, Disk2D, Polyline2D, Polypoint2D, Rectangle2D, TriangleSet2D

Chapter 13, Triangles and Quadrilaterals

- TriangleSet, TriangleStripSet, TriangleFanSet, QuadSet
- Both regular and Indexed versions

Advanced geometry nodes

Geospatial component

- GeoElevationGrid

NURBS component

- NurbsCurve, NurbsPatchSurface, NurbsSweptSurface, NurbsSwungSurface, NurbsTrimmedSurface

Programmable shaders component

- ComposedShader, PackagedShader, ProgramShader

Further information available in X3D Specification

- http://www.web3d.org/x3d/specifications

web|3D CONSORTIUM

Scalable Vector Graphics (SVG)

SVG is an XML language for two-dimensional (2D) graphics and graphical applications

- World Wide Web Consortium (W3C) Recommendations, working group
- http://www.w3.org/Graphics/SVG

Because both X3D and SVG are written in XML, we've created an XSLT stylesheet that makes SVG plots of 2D data structures in X3D scenes

- X3dExtrusionToSvgViaXslt1.0.xslt
- Linked via pretty-print html, example follows

web3D CONSORTIUM

76

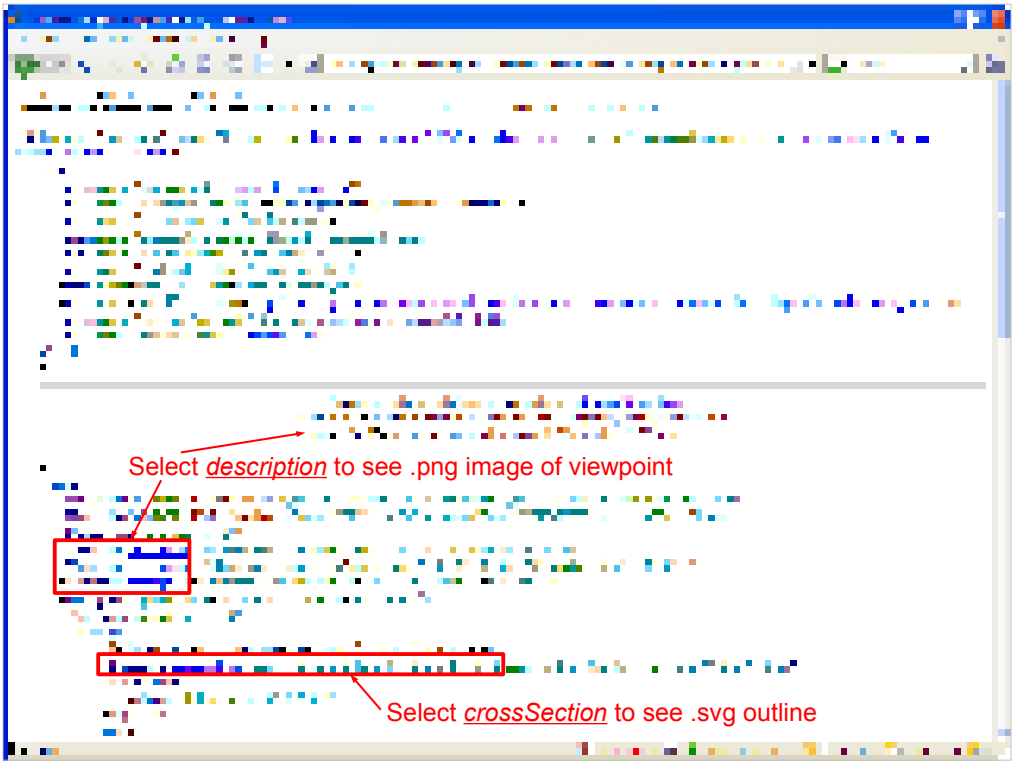
Stylesheets available via
http://x3d.svn.sourceforge.net/viewvc/x3d/www.web3d.org/x3d/stylesheets

The SVG autogenerated example works in the following viewers and browsers:

- Batik http://xmlgraphics.apache.org/batik
- Microsoft Internet Explorer 7
- Opera http://www.opera.com

... and partially works in the following web browsers:

- Firefox http://www.firefox.com
- W3C Amaya http://www.w3.org/Amaya



Pretty-print HTML

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ExtrusionPentagon.html

Viewpoint images

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/_viewpoints/ExtrusionPentagon.x3d.ExtrusionPentagon.png

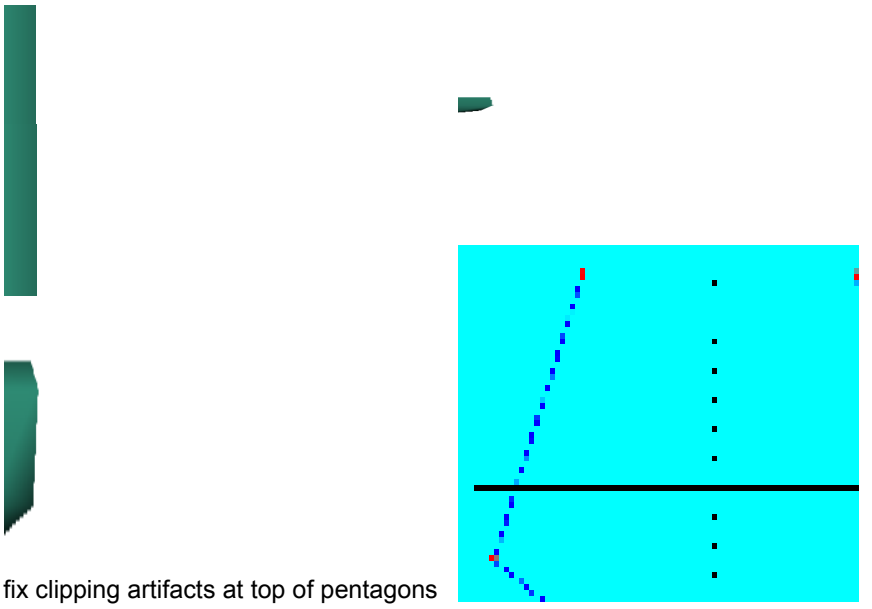
http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/_viewpoints/ExtrusionPentagon.x3d.Oblique_view_from_above.png

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/_viewpoints/ExtrusionPentagon.x3d.Overhead_view.png

Extrusion *crossSection* .svg diagram

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/_svg/ExtrusionPentagon.Extrusion1.svg

Viewpoint .png and Extrusion .svg output images



TODO: fix clipping artifacts at top of pentagons

Pretty-print HTML

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/ExtrusionPentagon.html

Viewpoint images

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/_viewpoints/ExtrusionPentagon.x3d.ExtrusionPentagon.png

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/_viewpoints/ExtrusionPentagon.x3d.Oblique_view_from_above.png

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/_viewpoints/ExtrusionPentagon.x3d.Overhead_view.png

Extrusion *crossSection* .svg diagram

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/_svg/ExtrusionPentagon.Extrusion1.svg

http://x3dgraphics.com/examples/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/_svg/ExtrusionPentagon.Extrusion1.png

back to Table of Contents

Chapter Summary

Chapter Summary 1

Polygonal geometry is essence of X3D graphics

- Also for most other computer graphics approaches

The rendering of almost every geometric shape is usually based on tessellation into triangles

Working with examples is the best way to learn.

Be patient, the principles are consistent, and practice helps make principles familiar.

web|3D CONSORTIUM

Chapter Summary 2

Triangles, single-sided polygons, normal vectors

Common fields: *ccw*, *convex*, *creaseAngle*, etc.

Geometry nodes, part 2

- Color and ColorRGBA
- Coordinate and CoordinateDouble
- PointSet
- IndexedLineSet and LineSet
- IndexedFaceSet
- ElevationGrid
- Extrusion

web3D CONSORTIUM

Suggested exercises

- Produce a simple graph of any sampled or functional X-Y data using IndexedLineSet
  - Also include axes and min/mix scale labels
- Write a simple program (in any language) that outputs coordinates for a circle's circumference
  - Insert outputs into a PointSet node for display
  - Show change when points are more closely spaced
- Build a simple object using an IndexedFaceSet
  - Add other IFS nodes with different Material values
- Build examples with ElevationGrid, Extrusion

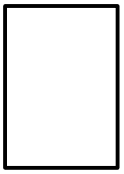
web|3D CONSORTIUM

back to Table of Contents

References

References 1

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



- Chapter 6, Geometry 2: Points Lines and Polygons
- http://x3dGraphics.com
- http://x3dgraphics.com/examples/X3dForWebAuthors

X3D Resources

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

web3D CONSORTIUM

References 2

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

web|3D CONSORTIUM

References 3

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

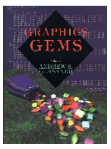
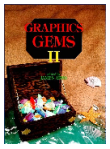
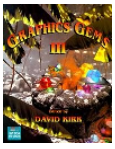
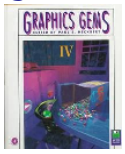
- http://www.wiley.com/legacy/compbooks/vrml2sbk/cover/cover.htm
- http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook
- Chapter 13 – Points Lines Faces
- Chapter 14 – Elevation Grid
- Chapter 15 – Extrusion
- Chapter 16 – Color

web3D CONSORTIUM

References 4

Graphics Gems book series: many algorithms

- http://www.graphicsgems.org


Journal of Graphics Tools (jgt): many algorithms

- http://jgt.akpeters.com

web 3D CONSORTIUM

X3D

87

http://www.merriam-webster.com/dictionary/algorithm

Term: al·go·rithm

Pronunciation: \ˈal-gə-ˌri-thəm\

Function: noun

Etymology: alteration of Middle English algorisme, from Old French & Medieval Latin; Old French, from Medieval Latin algorismus, from Arabic al-khuwārizmi, from al-Khwārizmī fl a.d. 825 Islamic mathematician

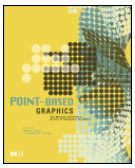
Date: 1926

Definition: a procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation; broadly : a step-by-step procedure for solving a problem or accomplishing some end especially by a computer

Note: a proper algorithm includes a termination condition.

References 5

Point-Based Graphics, Markus Gross and Hanspeter Pfister, editors, Morgan Kaufmann Publishers, 2007.



- http://www.elsevier.com/wps/find/bookdescription.cws_home/710117/description#description

Point-based Graphics Resources

- Ke-Sen Huang
- http://kesen.huang.googlepages.com/PointBasedPaper.html

web|3D CONSORTIUM

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

web|3D CONSORTIUM

89

CGEMS, SIGGRAPH, Eurographics

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

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

X3D for Web Authors recognized by CGEMS! ☺

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

CGEMS supported by SIGGRAPH, Eurographics


From the CGEMS home page:

- http://cgems.inesc.pt

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

List of all published modules:

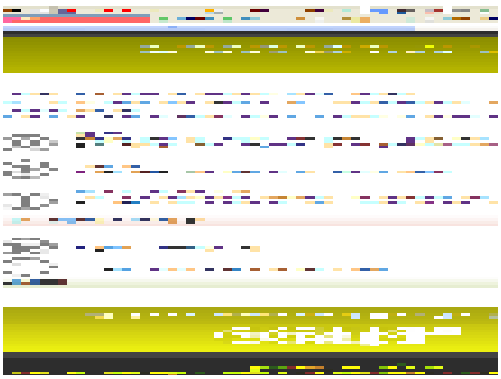
- http://cgems.inesc.pt/authors/ListModules.aspx

CGEMS Editorial Policy:

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

Creative Commons open-source license

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



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

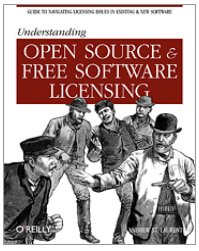
http://www.web3d.org/x3d/content/examples/license.html

License available at

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

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

Good references on open source:



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

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