X3D: an introduction

1 . **Technical Overview**. General introduction of the fundamentals of X3D, including scene graphs, events, node reuse, file structure and encodings, components and profiles, and conformance.

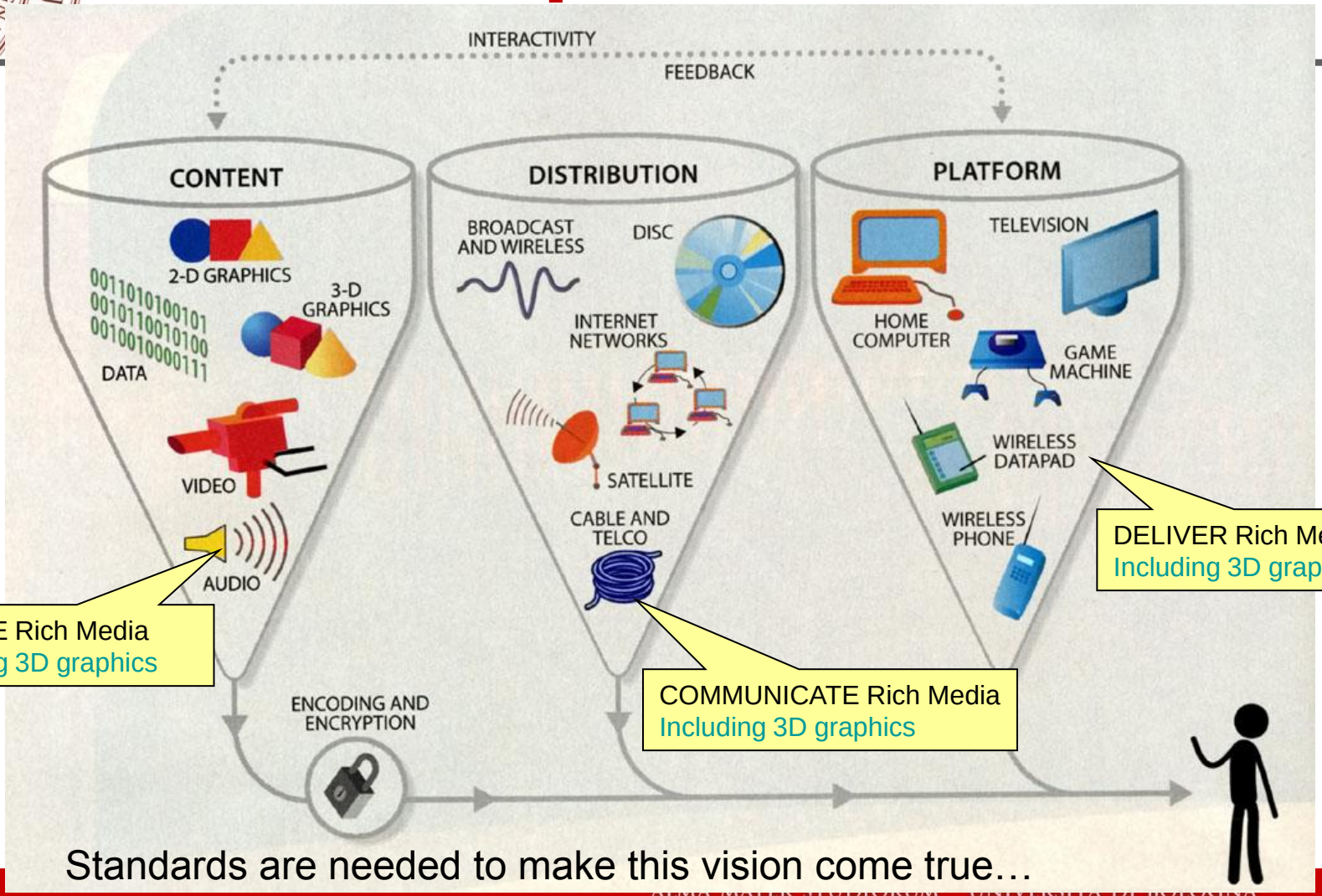
2. **X3D-Edit**

3. **Geometry.** Primitives, Polygons, NURBS

4. Viewpoints and Navigation

5. **Grouping and Transform Nodes**. Collecting and positioning objects in the 3D world.

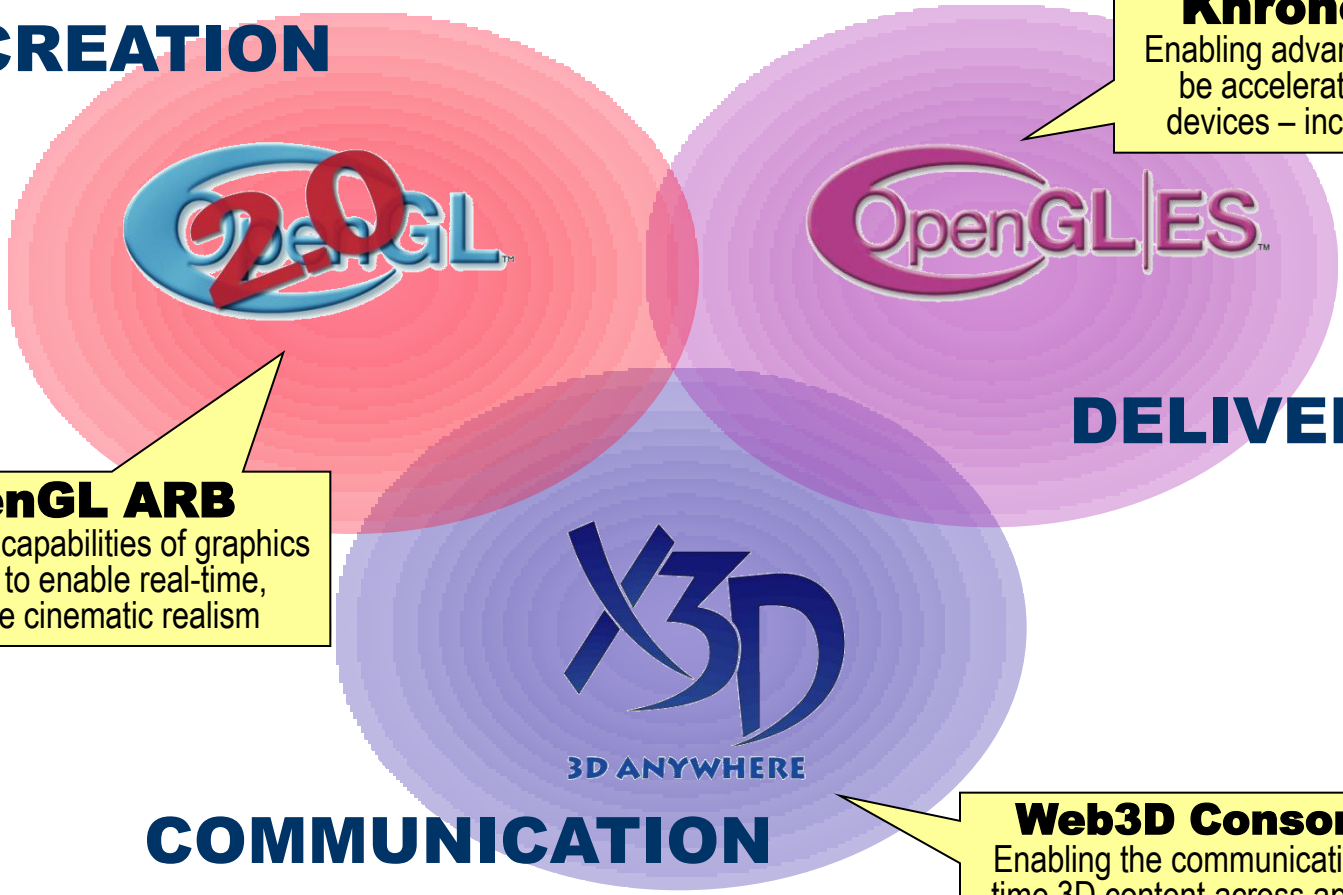
What are the pieces of the puzzle?



Standards are needed to make this vision come true…

web|3D CONSORTIUM

The Micro Universe of 3D Standards



Khronos Group
Enabling advanced 3D graphics to be accelerated on embedded devices – including cell phones

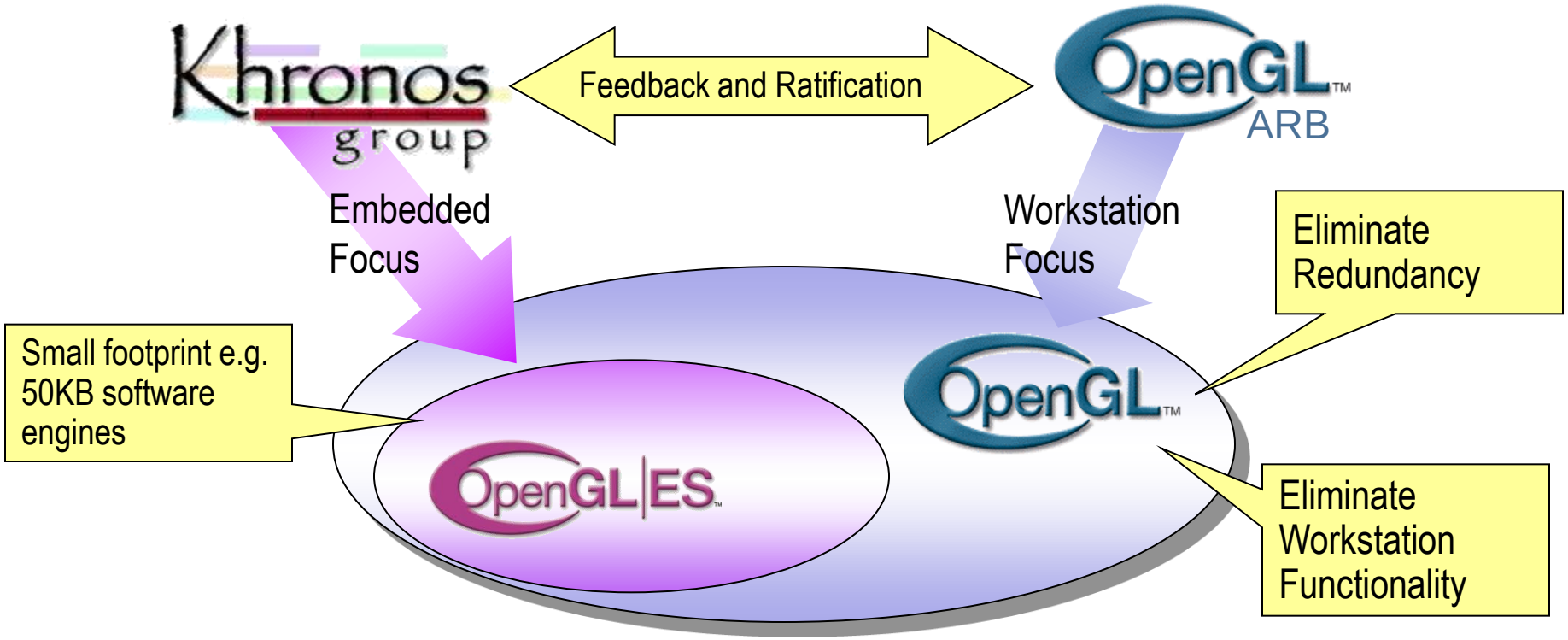
OpenGL ARB
Evolving the capabilities of graphics hardware to enable real-time, interactive cinematic realism

Web3D Consortium
Enabling the communication of real-time 3D content across applications, networks and the web

Note: not ignoring Java!
Java has a community standardization process
These technologies are used by Java applications

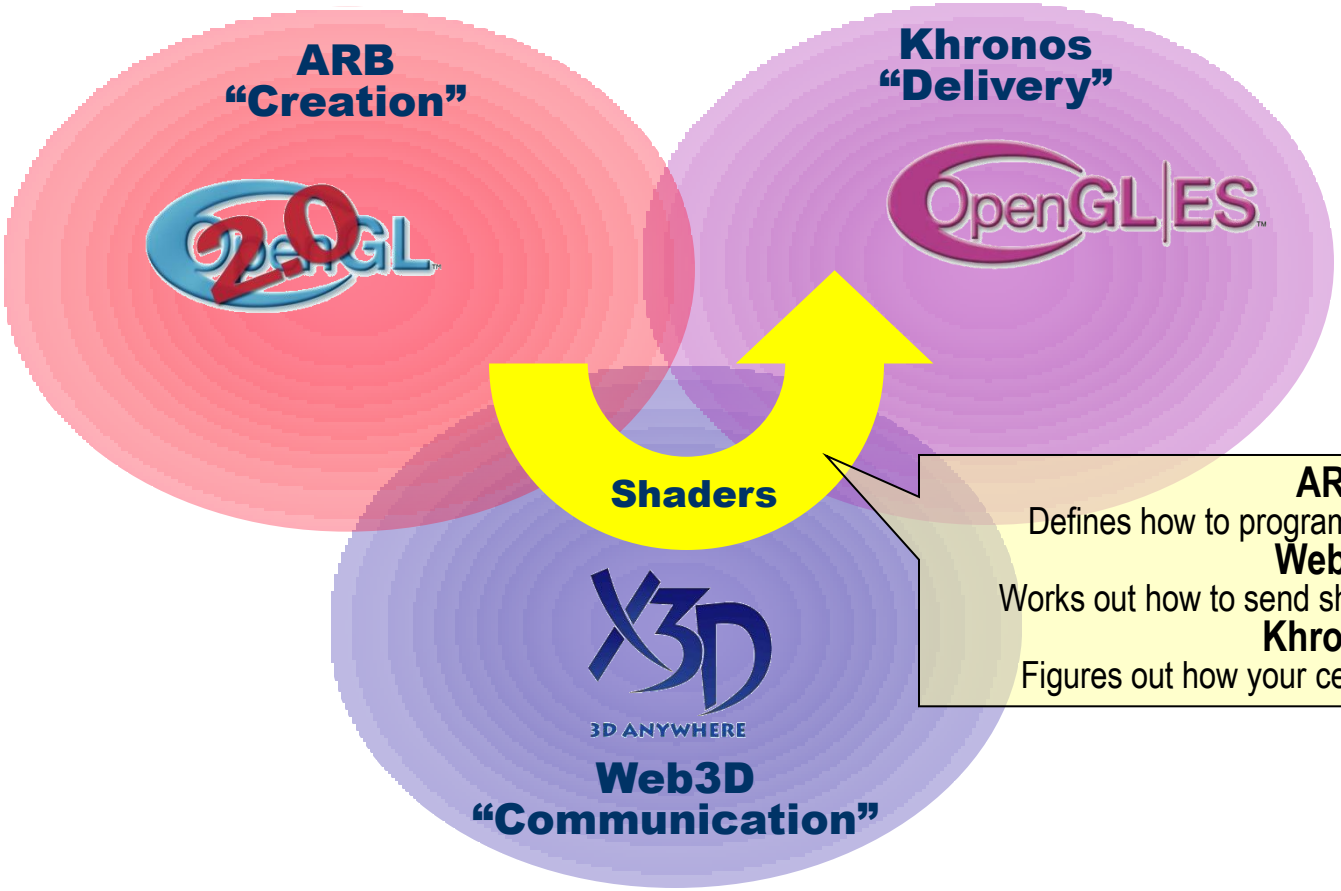
OpenGL ES – Embedded Graphics

- Khronos has created a small-footprint subset of OpenGL
  - Created with the blessing and cooperation of the OpenGL ARB
- Full functionality for 3D games
  - On a wide variety of platforms – including handhelds



web|3D CONSORTIUM

The Micro Universe of 3D Standards



ARB
Defines how to program shaders using GLSL
Web3D
Works out how to send shaders across a network
Khronos
Figures out how your cell phone will play them

web 3D CONSORTIUM

Communicating 3D is our Vital Role

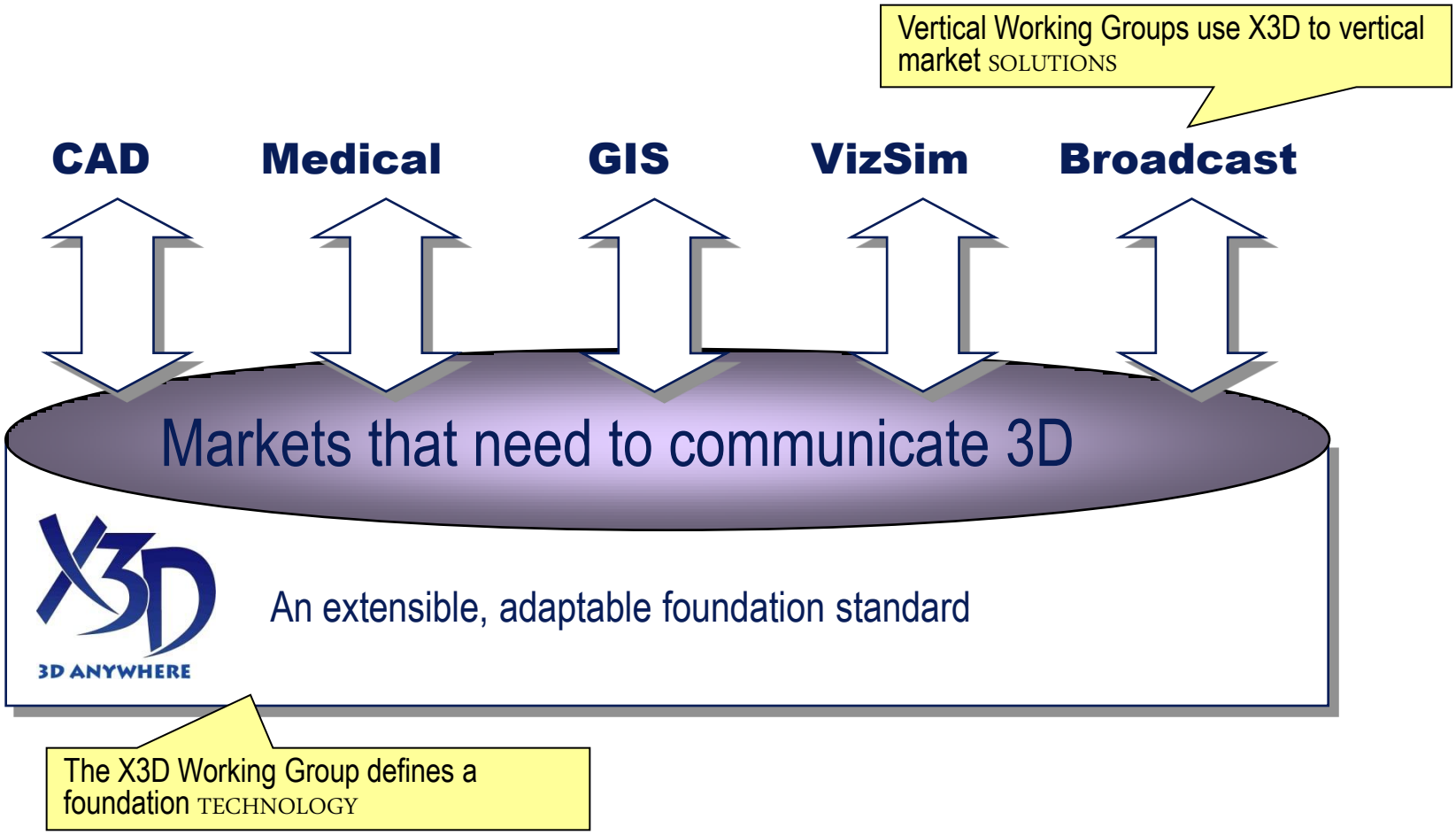
Between applications

"Open Standards to enable the communication of real-time 3D across networks
and XML-based web services"

Between systems

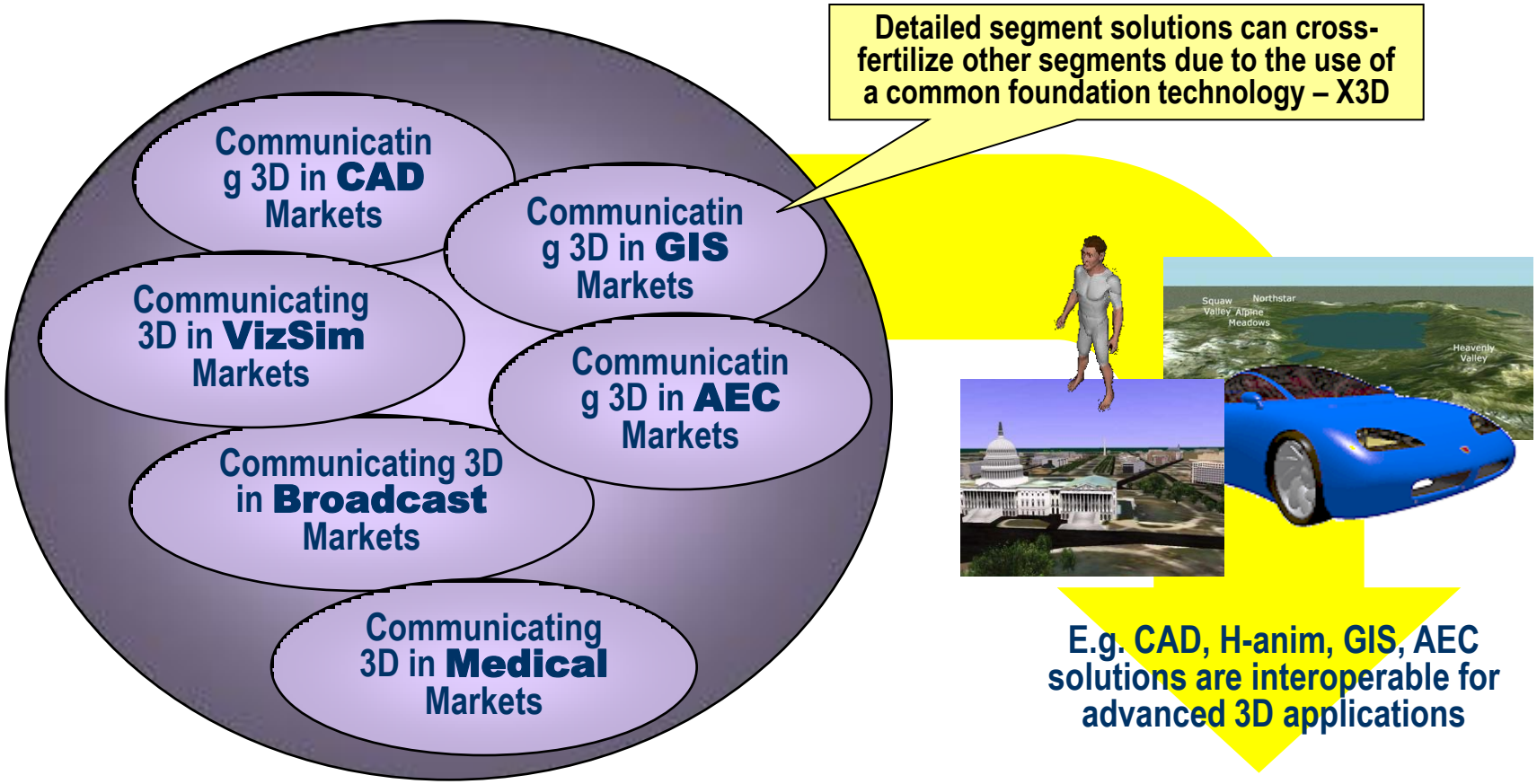
web|3D CONSORTIUM

X3D – a Trans-Segment Standard



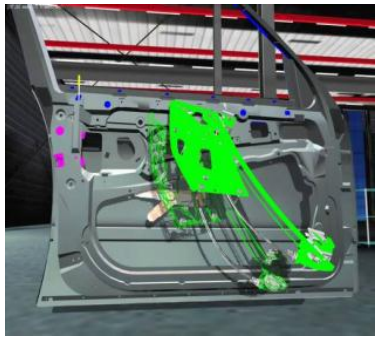
Cross Segment Synergy

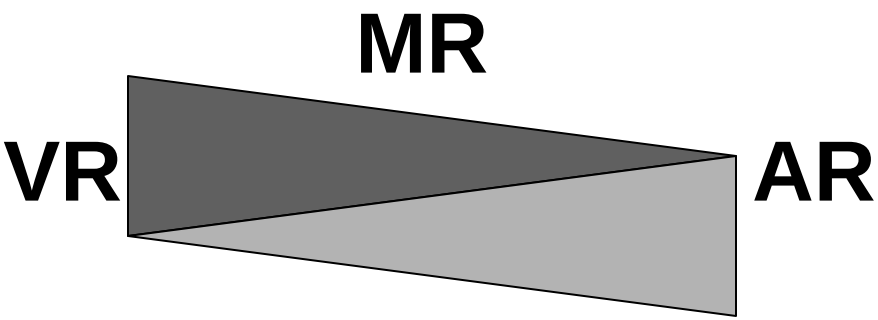
- Vertical focus is key to enable market segments
  - But a cross-segment ecosystem will begin to form to the benefit of all



Motivation to extend X3D for AR/MR

Virtual Objects





Real Objects

- Virtual Reality (VR) **– Virtual objects/ data**

  X3D is a well established application description language to express Virtual objects and their behaviors in 3D virtual environments

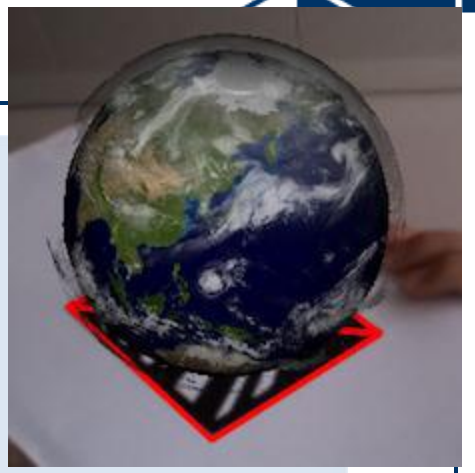
- Augmented Reality (AR) **– Virtual objects augmented by sensors**

  X3D has partial functionality for AR (Sensor Nodes, Viewpoint node, Camera node) Going beyond basic geo-location based AR

- Mixed Reality (MR) **– Continuum between VR and AR**

  X3D currently lacks features needed for MR - Extend X3D accommodate “real” world objects and represent MR contents

web|3D CONSORTIUM



Extending X3D for Augmented Reality

Fifth AR Standards Group Meeting

Anita Havele
Executive Director, Web3D Consortium
www.web3d.org
anita.havele@web3d.org
March 19, 2012

AR Working Group started in June 2011

Getting Started with X3D

A journey of a thousand miles
begins with a single step.
Chinese proverb

What is Extensible 3D (X3D)?

X3D is a royalty-free open-standard file format

- Communicate animated 3D scenes using XML
- Run-time architecture for consistent user interaction
- ISO-ratified standard for storage, retrieval and playback of real-time graphics content
- Enables real-time communication of 3D data across applications: archival publishing format for Web
- Rich set of componentized features for engineering and scientific visualization, CAD and architecture, medical visualization, training and simulation, multimedia, entertainment, education, and more

eXtensible 3D (X3D) Graphics

X3D is the International Organization for Standardization ISO standard XML-based file format for representing 3D virtual/augmented world in computer graphics

• The successor to the Virtual Reality Modeling Language (VRML). **VRML,** introduced in 1994, is the standard format to describe 3D contents for WEB, allows for combining 3D, 2D, text, video and audio;

• X3D features extensions to VRML (e.g. Humanoid animation, NURBS, GeoVRML etc.), integrates scripting functionalities and access to network resources.

• X3D file, multiple encoding. It can be coded in 3 different way:
  as an XML file (text file .x3d)
  as a VRML file (text file with VRML format .x3dv)
  as a binary file (by suitable conversion .x3db)

X3D: the Standard Scene graph

- An X3D file describes a 3D scene, a fully interactive 3D world, viewable on most standard internet browsers.

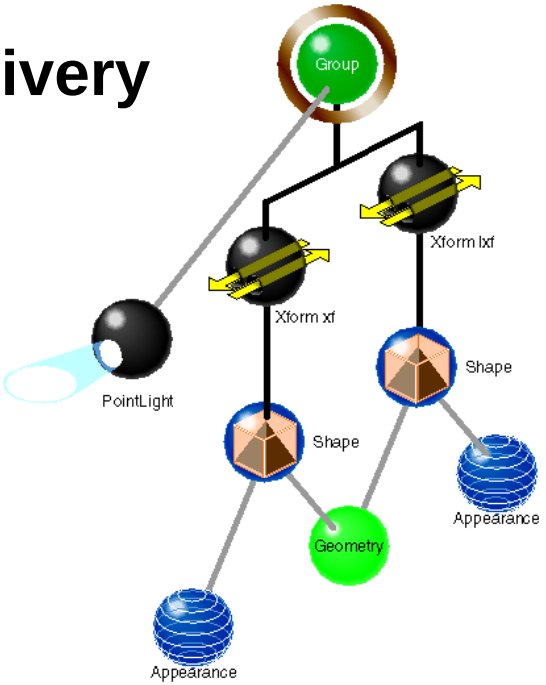
- X3D represents the 3D scene by means of a graph (**scenegraph**)
- Directed acyclic graph (DAG), meaning a tree with a root node and no loops

- Each aspect of a virtual world from simple 3D primitive shapes to lighting, animation and sound is considered a node. Each node will have various parameters describing how it behaves within the virtual world.

X3D: the Standard Scene graph

Scene graph for real-time interactive delivery of virtual environments over the web:

- Meshes, lights, materials, textures, shaders
- Integrated video, audio
- Animation
- Interaction
- Scripts & Behaviors



- Nearly all nodes have at least one input field and output field through which they may communicate with either the browser or another node.

- **X3D Version 3.3 in draft mode includes Volume rendering, CAD and Geospatial components.**

Scene graph terminology

Scene graph data file

- contains model description, may refer to data files

Scene graph viewer

- Reads and renders scene-graph models
- Implemented as application or web browser plugin

Scene graph editor

- Special text editor for scene graph development

Executable application

- Specific 3D model capable of running on a specific operating system

Scene graph rendering

The browser traverses the scene graph, updating any values within nodes and building an image

- New image then replaces previous screen image, process known as ***double buffering***
- Rapid repetitions are very important
- Frame rate faster than 7-10 Hz (cycles per second) provides appearance of smooth motion

Rendering is defined as this drawing process

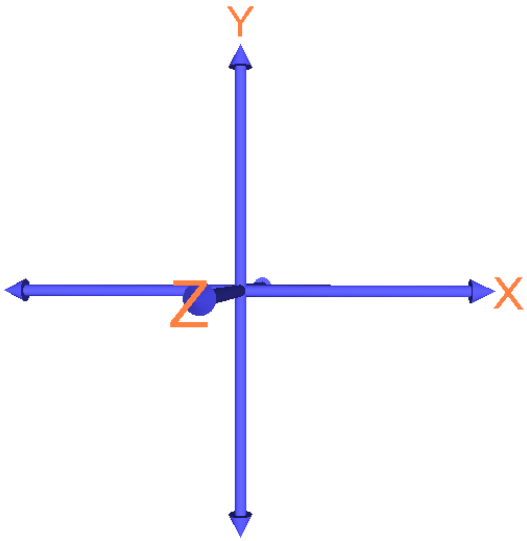
Off-line rendering is performing such operations to image or movie files, rather than display

Browser X3D

- Applications able to read an X3D file and visualize it, also reproducing dynamical aspects, and interacting with the user
- In general, they are plug-in for Web browser; thus we have:
  - X3D file can be loaded and visualized by URL
  - X3D file can be combined with other Web page contexts

Scene Visualization (view)

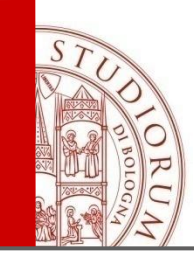
- Users explore X3D worlds by choosing predefined viewpoints and navigating through 3D space.
- Each point in the world is identified by real coords. (x,y,z) w.r.t. a global coords. system with axes X, Y e Z
- Right hand rule for X Y Z order



- The user is represented inside the world by his/her **avatar**
- **The scene is seen by the camera located on the (ideal) avatar's head**

How to move into the scene

- **Navigation mode:**
  - **WALK:** include gravity force (siutable for walking inside a house)
  - **FLY:** no gravity (suitable for moving in the space)
  - **EXAMINE:** examine specific objects
  - ....
- Each class have several options to control the position and view orientation
- The movement is obtained by keypress (es. arrows) or mouse (generally, left mouse click)



Examine navigation controls:

Mouse:

Rotate	Left mouse dragging
Move	Middle mouse dragging (or Left mouse + Shift)
Zoom	Right mouse dragging (or Left mouse + Ctrl)

Keys:

Rotate	Arrows / PageUp / PageDown
Stop rotating	Space
Move	Ctrl + Arrows / PageUp / PageDown
Scale	+ / -
Restore default transformation	Home

Walk / Fly navigation controls:

Basic:

Forward / backward	Up / Down
Rotate	Left / Right
Raise / bow your head	PageUp / PageDown
Restore head raise to initial position (neutralize any effect of PageUp / PageDown)	Home
Fly up / down	Insert / Delete
Move left / right	Comma / Period
Jump / crouch (only when Gravity works, in Walk mode)	A / Z

Turn "Mouse Look" "On" (Ctrl+M) to comfortably look around by moving the mouse.
In the "Mouse Look" mode, the keys for strafe and rotations swap their meaning:

- Left / Right keys move left / right
- Comma / Period rotate

Additional controls:

Increase / decrease moving speed	+ / -
Increase / decrease avatar height	Ctrl + Insert/Delete
Rotate slower	Ctrl + Left / Right
Raise / bow your head slower	Ctrl + PageUp / PageDown
Pick a point, selecting triangle and object	Right mouse click

X3D file structure

X3D scene files have a common file structure

- File header (XML, ClassicVRML, Compressed Binary)

- X3D header statement
  - Profile statement
  - Component statements (optional)
  - Meta statements (optional)

- X3D root node
- X3D scene graph child nodes

Header XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN" "http://www.web3d.org/specifications/x3d-3.2.dtd">
<X3D profile='Immersive' version='3.2' >
  <head>
```

Header X3D

```
    <meta content='HelloWorld.x3d' name='title'/>
    <meta content='Simple X3D example' name='description'/>
  </head>
```

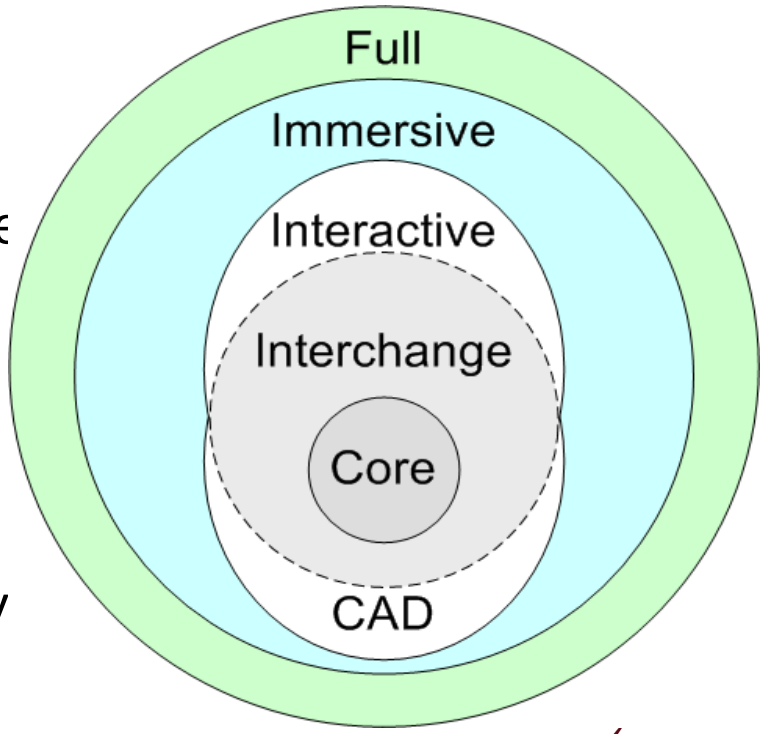
Scenegraph

```
  <Scene>
    <Viewpoint description='Hello world!' position='0 -1 7'/>
      <Shape>
        <Sphere/>
        <Appearance>
        <Material DEF='LightBlue' diffuseColor='0.1 0.5 1'/>
        </Appearance>
      </Shape>
  </Scene>
</X3D>
```

Profiles cover common use cases

Authors define the expected complexity of scene by defining profile level in the X3D header. This tells the X3D browser what level of functional support is needed for run-time operation

- *Interchange* suitable for simple geometry conversion
- *Interactive* adds simple user interactivity (clicking etc.)
- *Immersive* matches VRML97, plus a bit more "implementing immersive virtual worlds with complete navigational and environmental sensor control"
- *Full* profile includes all nodes



Further customization within a scene is alway possible using **component** statements to identify the correct level of functional support beyond the identified profile.

meta statements

meta statements provide information about the X3D scene

- Document metadata, not scene metadata

Information provided as name-value pairs

- Example:

```
<meta name='created' value='1 January 2008'/>
```

This approach is thus very general

- Wide variety of metadata can be represented
- Matches same approach used by HTML for regular hypertext web pages

profile, component and meta statements, XML (.x3d) encoding syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN" "http://www.web3d.org/specifications/x3d-3.2.dtd">
<X3D version="3.2" profile="Immersive" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
                    xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.2.xsd">
    <head>
        <component name='DIS' level='1'/>
        <component name='Geospatial' level='1'/>
        <component name='H-Anim' level='1'/>
        <component name='NURBS' level='4'/>
        <meta name='title' content='HeaderProfileComponentMetaExample.x3d'/>
    </head>
    <Scene>
        <!——Scene graph nodes are added here——>
    </Scene>
</X3D>
```

XML and X3D correspondence

Opening element	`<Shape>`
Singleton element, attribute="value"	`<Sphere radius="10.0" solid="true"/>`
Opening element	`<Appearance>`
Singleton element, attribute='value'	`<ImageTexture url='earth-topo.png'/>`
Closing element	`</Appearance>`
Closing element	`</Shape>`

Elements correspond to X3D **nodes**

Attributes correspond to X3D simple-type **fields**

Parent-child relationships define **containerField**

Validatable XML using X3D DTD, schema

The Extensible Markup Language (XML) is a plain-text format used by many Web languages including Hypertext Markup Language (HTML)

Suggested Exercise 1: getting start..

- Copy the X3D file into a text file; save it as a .x3d file;
- Visualize the scene open it by browser (i.e. **`viewer3dscene`**)

  and try to use the different modalities to navigate into the scene
- Add the code ->

  and run

```
<Scene>
  <Group>
  <Viewpoint description='Hello world!' position='0 -1 7'/>
   <Shape>
      <Sphere/>
      <Appearance>
      <Material DEF='LightBlue' diffuseColor='0.1 0.5 1'/>
      </Appearance>
   </Shape>
            <Transform translation='0 -2 0'>
    <Shape>
      <Text string='“Hello" "world!"'>
       <FontStyle justify='"MIDDLE" "MIDDLE"'/>
      </Text>
      <Appearance>
       <Material USE='LightBlue'/>
      </Appearance>
    </Shape>
   </Transform>
  </Group>
 </Scene>
```

X3D-Edit authoring tool

Software support – it supports the creation, checking, display and publication of X3D scenes.

X3D-Edit

Available free for any use

- https://savage.nps.edu/X3D-Edit
- Written using Java, XML and X3D
- Windows, MacOSX, Linux, Solaris operating systems

Standalone application with automatic updates available once installed

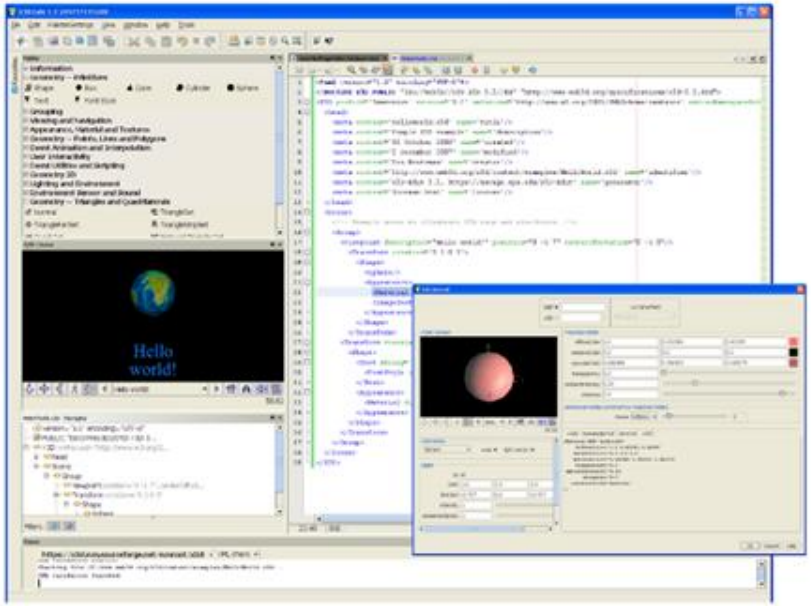
Also available for Netbeans as plugin module

- Open integrated development environment (IDE), primarily (but not exclusively) for Java
- http://www.netbeans.org

X3D-Edit Authoring Tool for Extensible 3D (X3D) Graphics

web3D CONSORTIUM
Open Standards for Real-Time 3D Communication

X3D-Edit is an Extensible 3D (X3D) Graphics authoring tool for simple error-free editing, authoring and validation of X3D scenes.



Overview

The X3D-Edit 3.2 Authoring Tool for Extensible 3D (X3D) Graphics supports the creation, checking, display and publication of X3D scenes. It is written in open-source Java and XML using the Netbeans platform, making it suitable both as a standalone application and as a plugin module for the Netbeans integrated development environment (IDE).

X3D-Edit features include direct editing of X3D scenes using the XML (.x3d) encoding, embedded visualization of scenes using the Xj3D viewer, XML validation against X3D DTDs and Schemas, drag-and-drop palette for X3D nodes, popup panels for node editing, and extensive help resources. Planned features include ClassicVRML and X3D compressed binary encoding support, encryption and digital-signature authentication using XML Security standards, and additional X3D scene authoring support.

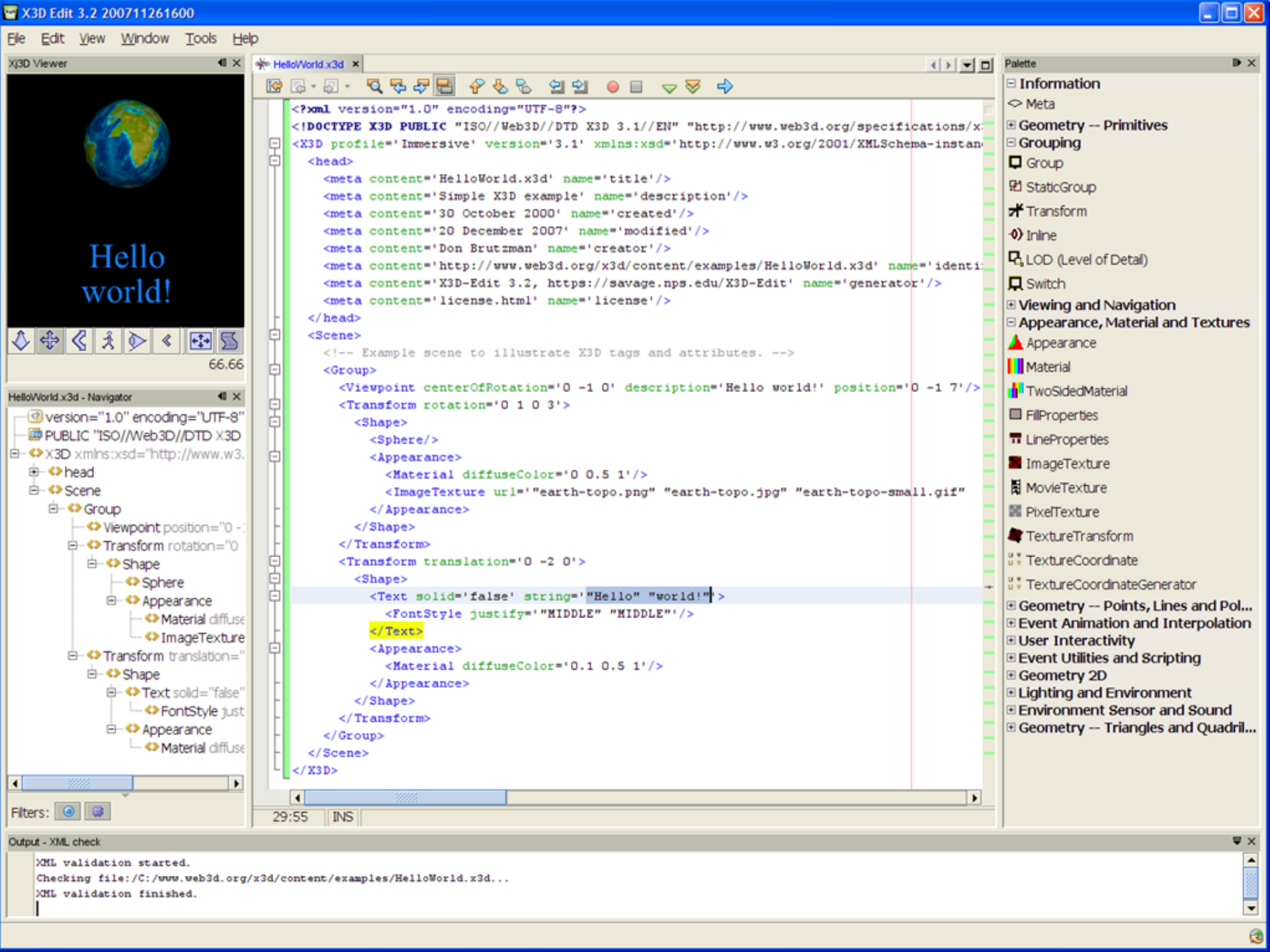
X3D-Edit download and installation

Options on X3D-Edit home page

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

Standalone executable application:

- Download and extract X3D-Edit3.2.zip
- https://savage.nps.edu/X3D-Edit/X3D-Edit3.2.zip
- Launch *runX3dEditWin.bat* on a Windows machine
- Launch *runX3dEditMac.sh.command* on a Mac
- Successful test reports received for Linux...
- That's all there is to it!


X3D Edit 3.2 200711261600
File Edit View Window Tools Help
X3D Viewer
Hello world!
66.66
HelloWorld.x3d - Navigator
version="1.0" encoding="UTF-8"
PUBLIC "ISO//Web3D//DTD X3D
X3D xmlns:xsd="http://www.w3.
head
Scene
Group
Viewpoint position="0 -
Transform rotation="0
Shape
Sphere
Appearance
Material diffuse
ImageTexture
Transform translation="
Shape
Text solid="false"
FontStyle just
Appearance
Material diffuse
Filters:
HelloWorld.x3d
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instan
<head>
<meta content='HelloWorld.x3d' name='title'/>
<meta content='Simple X3D example' name='description'/>
<meta content='30 October 2000' name='created'/>
<meta content='20 December 2007' name='modified'/>
<meta content='Don Brutzman' name='creator'/>
<meta content='http://www.web3d.org/x3d/content/examples/HelloWorld.x3d' name='identi
<meta content='X3D-Edit 3.2, https://savage.nps.edu/X3D-Edit' name='generator'/>
<meta content='license.html' name='license'/>
</head>
<Scene>
<!-- Example scene to illustrate X3D tags and attributes. -->
<Group>
<Viewpoint centerOfRotation='0 -1 0' description='Hello world!' position='0 -1 7'/>
<Transform rotation='0 1 0 3'>
<Shape>
<Sphere/>
<Appearance>
<Material diffuseColor='0 0.5 1'/>
<ImageTexture url='"earth-topo.png" "earth-topo.jpg" "earth-topo-small.gif"
</Appearance>
</Shape>
</Transform>
<Transform translation='0 -2 0'>
<Shape>
<Text solid='false' string='"Hello" "world!"'>
<FontStyle justify='"MIDDLE" "MIDDLE"'/>
</Text>
<Appearance>
<Material diffuseColor='0.1 0.5 1'/>
</Appearance>
</Shape>
</Transform>
</Group>
</Scene>
</X3D>
29:55
INS
Palette
Information
Meta
Geometry -- Primitives
Grouping
Group
StaticGroup
Transform
Inline
LOD (Level of Detail)
Switch
Viewing and Navigation
Appearance, Material and Textures
Appearance
Material
TwoSidedMaterial
FillProperties
LineProperties
ImageTexture
MovieTexture
PixelTexture
TextureTransform
TextureCoordinate
TextureCoordinateGenerator
Geometry -- Points, Lines and Pol...
Event Animation and Interpolation
User Interactivity
Event Utilities and Scripting
Geometry 2D
Lighting and Environment
Environment Sensor and Sound
Geometry -- Triangles and Quadril...
Output - XML check
XML validation started.
Checking file:/C:/www.web3d.org/x3d/content/examples/HelloWorld.x3d...
XML validation finished.

Field data types

X3D is a strongly typed language

- Each field in each node (i.e. each XML attribute) has a strictly defined data type
- Data types for boolean, integer, floating point

Types are either single or multiple-value

- Example: SFFloat, SFVec2f, SFVec3f, SFOrientation

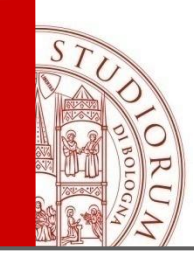
Also have arrays for all types

SF = Single Field, MF = Multiple Field (array)

MF are surrounded by square brackets, e.g. `[ 10 20 30, 4.4 -5.5 6.6 ]`

Failure to match data types correctly is an error!

- During scene validation, loading, or at run time



Viewing and Navigation

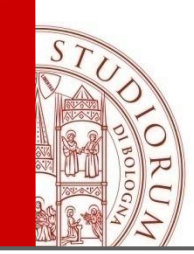
Viewing and navigation

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

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

Making navigation easy for users is important

- Authors provide viewpoints of interest with scenes
- Browsers enable camera rotation, pan, zoom, etc.



Viewpoint node

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

Default Viewpoint *position* is (0 0 10) -- out 10 m on +Z axis, looking back towards origin. Any changes to Viewpoint *orientation* are made relative to that default direction (along -Z axis)

Sometimes viewpoints are animated and moving

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

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

KelpForestMain.x3d (model/x3d+xml Object) - Mozilla Firefox

File Edit View History Bookmarks Tools GUtil! Help

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

john lennon acoustic

X3D Specification, ISO/IEC 1... | X3D for Web Authors, Kelp F... | Help: X3D Examples | X3D Scene Authoring Hints | KelpForestMain.x3d (mo...

Welcome to the NPS simulation of the Monterey Bay Aquarium Kelp Forest

Find sharks! See new viewpoints!
Press PageDown, wait and watch.

explore examine seek level back forward <- Kelp Forest Monter ->

Find: toolti Next Previous Highlight all Match case

Done 1.953s 72.52.156.96 70.134.87.160 0:906 Now: Rain, 51° F Sun: 55° F Mon: 54° F

Navigation model 1

Users can select predefined Viewpoints

- Defines both position and direction of view

Users can further navigate around scene

- Using pointing device or hot keys
- Chosen viewpoint remains bound

Key	Emulated Action	WALK mode	FLY mode	EXAMINE mode
Up arrow	Pointer up	forward	forward	orbit up
Down arrow	Pointer down	backward	backward	orbit down
Left arrow	Pointer left	left	left	orbit left
Right arrow	Pointer right	right	right	orbit right

These are the default navigation key responses

Navigation model 2

User's current view can itself be animated

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

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

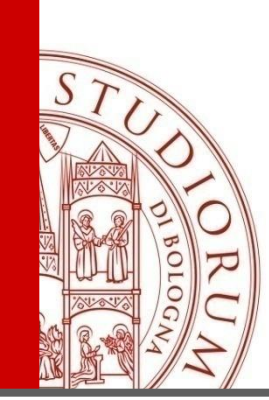
NavigationInfo *type*

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

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

MFString array default *type*=' "EXAMINE" "ANY" '

- which gives users plenty of flexibility

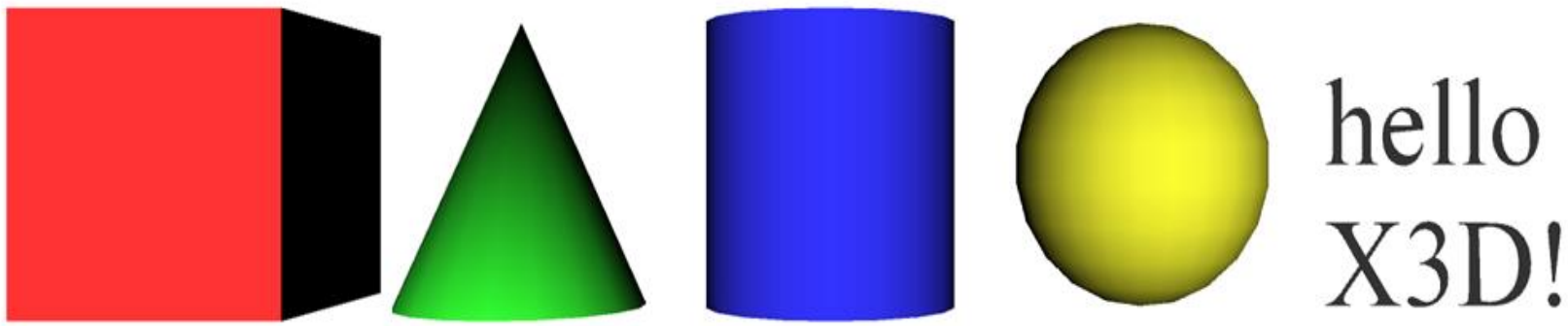


Geometry:

1. **Primitive Shapes**
2. Points, Lines and Polygons
3. Geometry2D Nodes
4. Triangles and Quadrilaterals
5. NURBS

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

Geometry 1, Primitive Shapes



Common pattern for Shape nodes

–Shape contains geometry node

–Appearance and Material nodes

Five nodes for primitive geometry

– Box, Cone, Cylinder, Sphere, Text

– Text node is flat, not extruded

– Font Style modifies Text node parameters

Shape and geometry

Shape nodes can contain a single geometry node

- For example, one of the five geometry primitive nodes
- Alternatively contains a more-advanced geometry node (NURBS, Geospatial, programmable shaders,..)

Shape nodes can also contain an Appearance node

- Which in turn contains a Material node for coloring

- Common design pattern throughout X3D:
  - Shape
    - GeometryNode
    - Appearance
      - Material (optional) for colors
      - ImageTexture (optional) for wrapping an image file

Shape parent with geometry child

```
<Shape>
    <Box size='1 2 3'/>
    <Appearance>
<Material/>
    </Appearance>
</Shape>
```

Shape must be parent node, can only hold one geometry node

Appearance and Material nodes define colors, transparency, etc.

```
<Shape>
    <Sphere radius='1'/>
    <Appearance>
<Material/>
    </Appearance>
</Shape>
```

Primitives have simple dimensions

- Typical volume ~1 m radius

All units are in meters

Note parent-child relationships

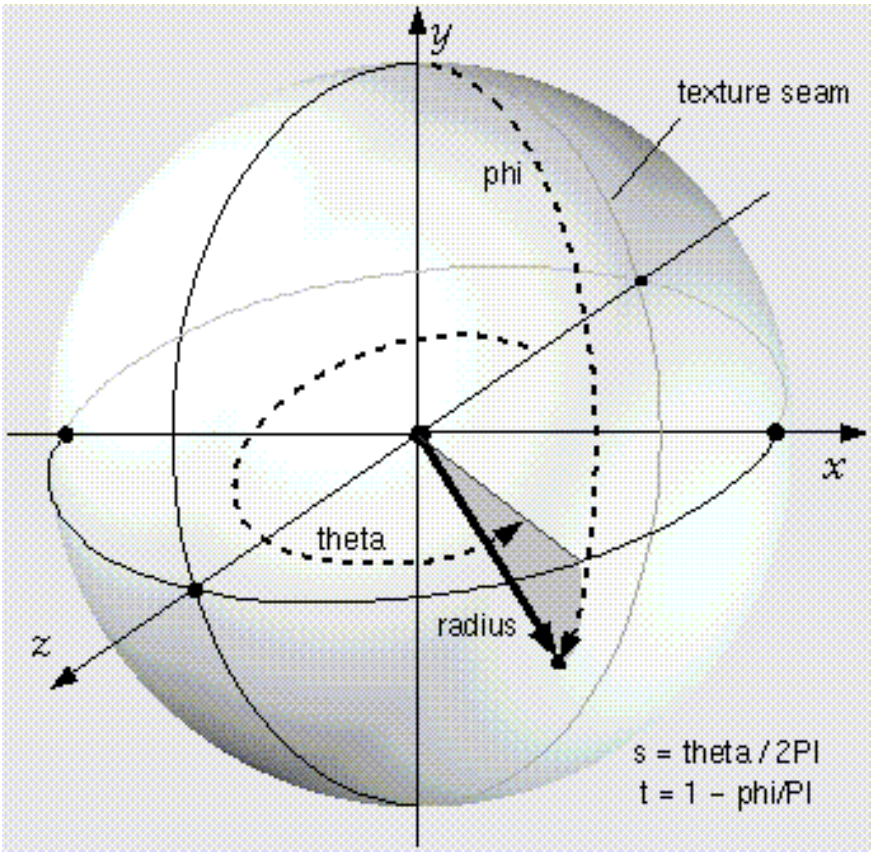
Sphere node

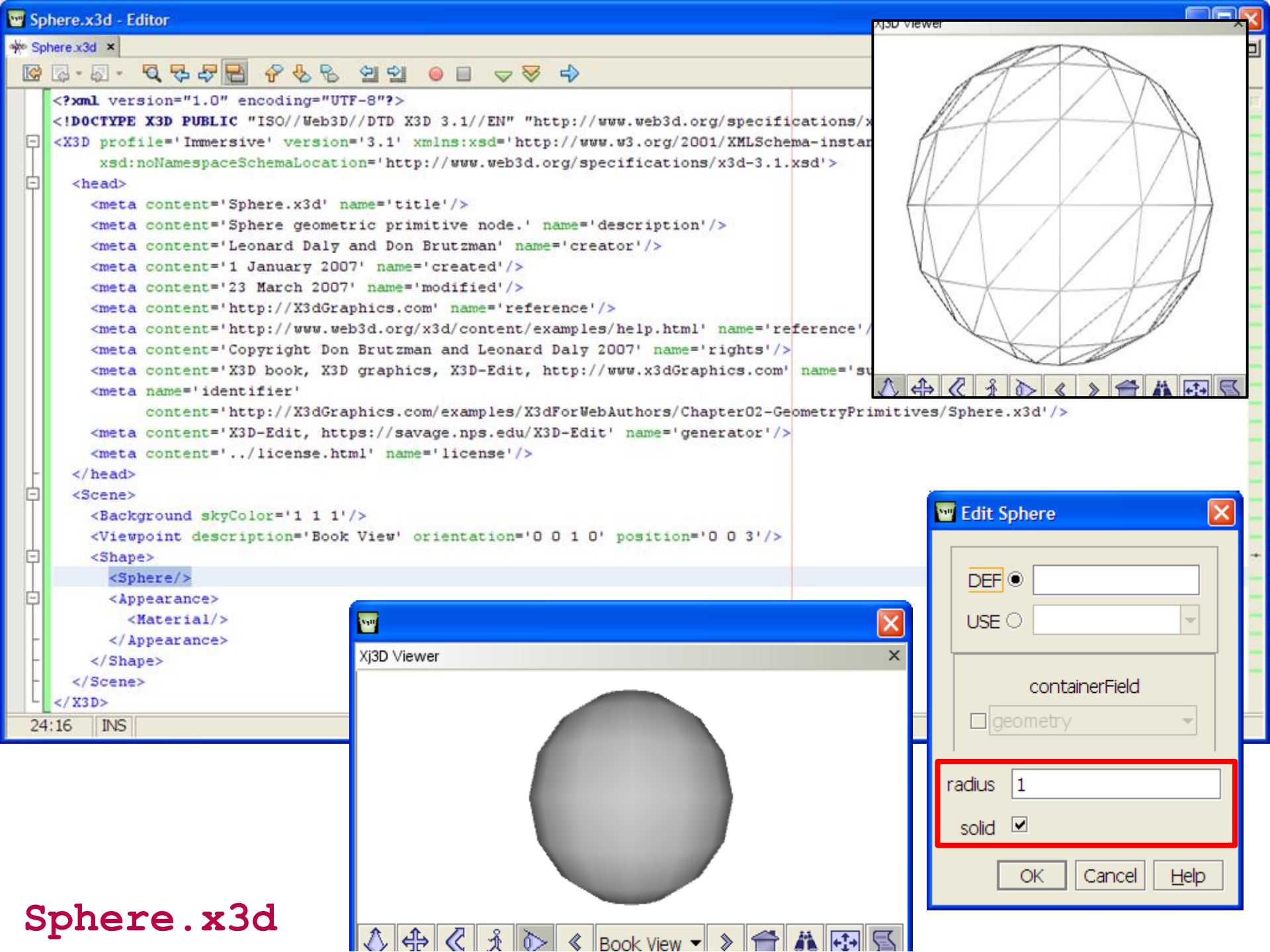
Circular *radius*

Centered at local origin

- phi and theta are implicit
- not defined by author

- Browsers decide implementation details, including tessellation (polygon count) and thus quality




Sphere.x3d - Editor
Sphere.x3d
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instan
xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.1.xsd'>
<head>
<meta content='Sphere.x3d' name='title'/>
<meta content='Sphere geometric primitive node.' name='description'/>
<meta content='Leonard Daly and Don Brutzman' name='creator'/>
<meta content='1 January 2007' name='created'/>
<meta content='23 March 2007' name='modified'/>
<meta content='http://X3dGraphics.com' name='reference'/>
<meta content='http://www.web3d.org/x3d/content/examples/help.html' name='reference'/
<meta content='Copyright Don Brutzman and Leonard Daly 2007' name='rights'/>
<meta content='X3D book, X3D graphics, X3D-Edit, http://www.x3dGraphics.com' name='su
<meta name='identifier'
content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter02-GeometryPrimitives/Sphere.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='Book View' orientation='0 0 1 0' position='0 0 3'/>
<Shape>
<Sphere/>
<Appearance>
<Material/>
</Appearance>
</Shape>
</Scene>
</X3D>
24:16
INS
Xj3D Viewer
Xj3D Viewer
Book View
Edit Sphere
DEF
USE
containerField
geometry
radius
1
solid
OK
Cancel
Help
Sphere.x3d

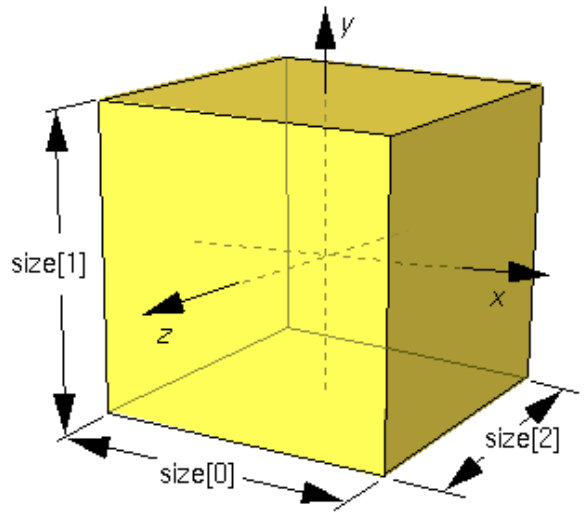
X3D Tooltips - Mozilla Firefox

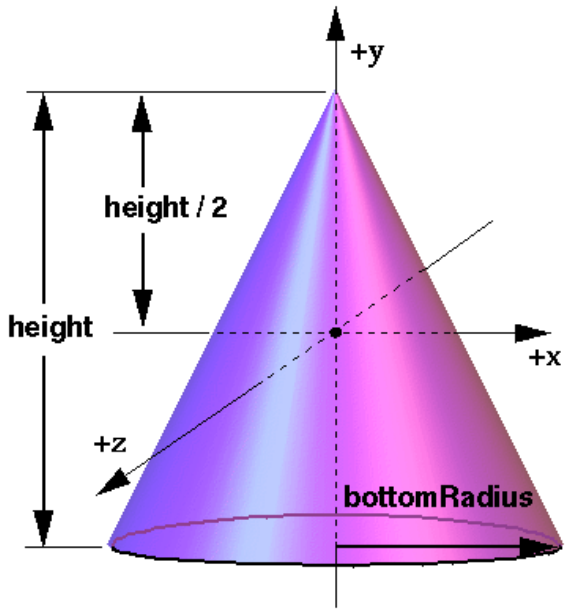
File Edit View History Bookmarks Tools GUtil Help

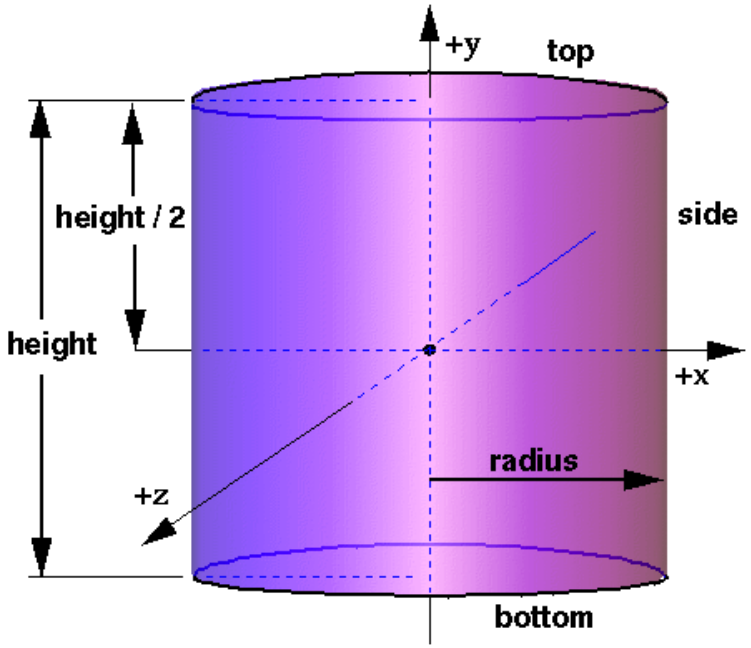
http://www.web3d.org/x3d/content/X3dTooltips.html#Sphere

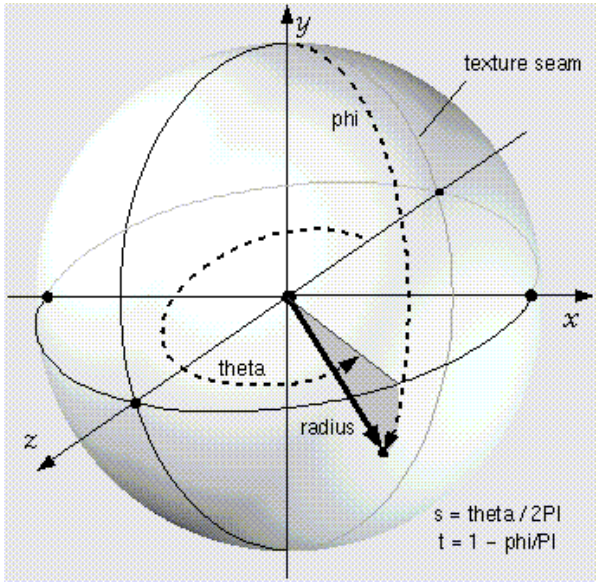
	top help credits
Sphere	**Sphere is a geometry node.** Hint: **insert a Shape node before adding geometry or Appearance.**
DEF	**[DEF ID #IMPLIED]** DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model.
USE	**[USE 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!
radius	**[radius: accessType initializeOnly, type SFFloat CDATA "1"]** Size in meters. Warning: simple-geometry dimensions cannot be changed after initial creation, use Transform scale instead.
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! Warning: solid false not supported in VRML97.
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]

•X3D Specification Diagrams









back to Table of Contents

Grouping and Transformation

Grouping rationale

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

Grouping nodes help provide sensible structure

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

Bounding boxes

Provides a hint to browsers about object size

- Does not affect how an object is rendered (drawn) if it is actually larger than the bounding box
- Are never drawn themselves
- Defined by *bboxSize* and *bboxCenter*

Goal is to reduce computational complexity

- browser avoids calculating impossible collisions
- Size accumulates while proceeding up scene graph

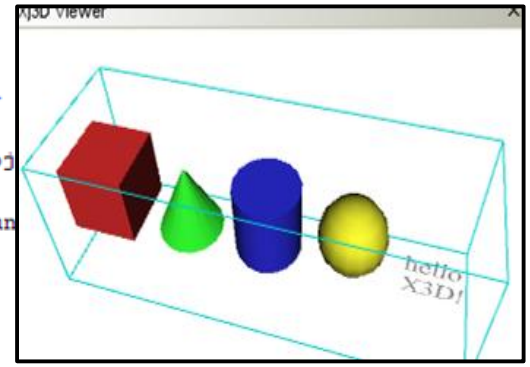
Bounding boxes can be ignored by authors

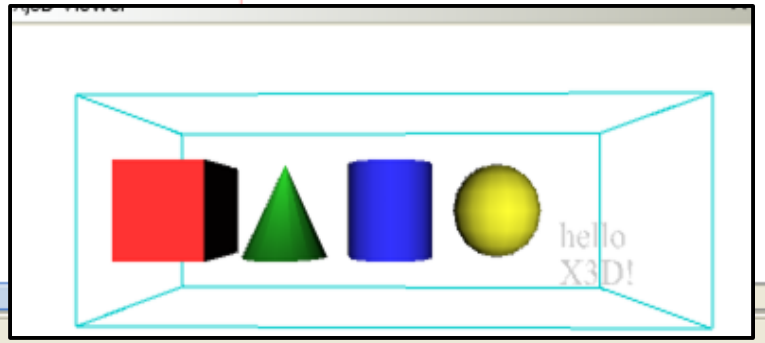
- some authoring tools can provide them if needed

BoundingBoxIllustration.x3d - Editor

BoundingBoxIllustration.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='BoundingBoxIllustration.x3d' name='title'/>
    <meta content='Simple Inline example illustrating bounding box coverage.  Bounding box lines are not typically rendered.' name
    <meta content='Don Brutzman' name='creator'/>
    <meta content='28 December 2005' name='created'/>
    <meta content='28 December 2007' name='modified'/>
    <meta content='http://X3dGraphics.com' name='reference'/>
    <meta content='http://www.web3d.org/x3d/content/examples/help.html' name='reference'/>
    <meta content='Copyright Don Brutzman and Leonard Daly 2007' name='rights'/>
    <meta content='X3D book, X3D graphics, X3D-Edit, http://www.x3dGraphics.com' name='subj
    <meta name='identifier'
          content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter03-Grouping/Boun
    <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="Bounding box illustration" position="0 0 15" fieldOfView="0.785"/>
    <Group bboxSize="12 4 4">
        <Inline url='"../Chapter02-GeometryPrimitives/GeometryPrimitiveNodes.x3d"
                    "../Chapter02-GeometryPrimitives/GeometryPrimitiveNodes.wrl"
                    "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter02-GeometryPrimitives/GeometryPrimitiveNodes.x3d"
                    "http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter02-GeometryPrimitives/GeometryPrimitiveNodes.wrl"'/>
        <Shape>
            <IndexedLineSet coordIndex="0 1 2 3 0 -1, 4 5 6 7 4 -1, 0 4 -1, 1 5 -1, 2 6 -1, 3 7 -1">
                <Coordinate point="-6 -2 -2, -6 -2 2, 6 -2 2, 6 -2 -2, -6 2 -2, -6 2 2, 6 2 2, 6 2 -2"/>
            </IndexedLineSet>
            <Appearance>
                <!-- lines are only lit by emissiveColor -->
                <Material emissiveColor="0 0.8 0.8"/>
            </Appearance>
        </Shape>
    </Group>
  </Scene>
</X3D>
```





23:30 INS

Transform node

Grouping node that defines a coordinate system for its children

Root of X3D scene graph is always at (0 0 0)

Transform nodes can

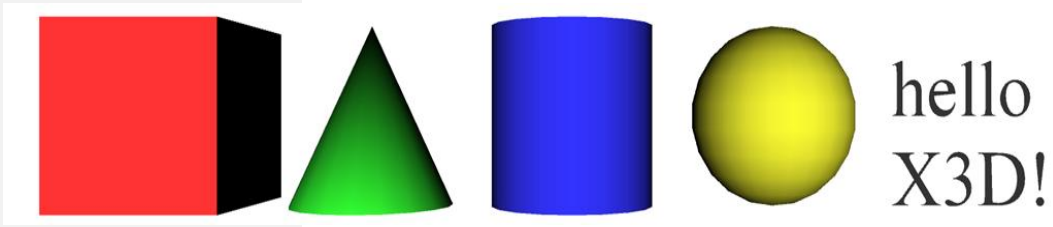
- Translate local origin linearly to another coordinate
- Rotate about any axis
- Scale size, uniformly or separately along x y z axes

Group and Transform are among most commonly used nodes

Transform fields

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

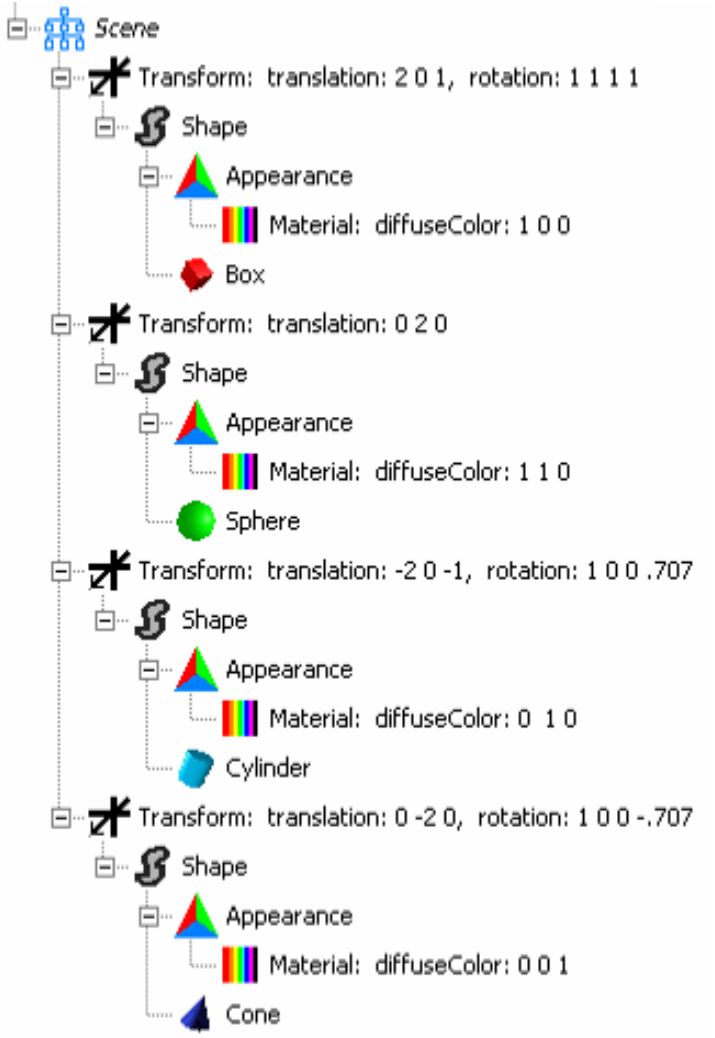
```
<Scene>
  <Transform translation='-5 0 0'>
   <Shape DEF='DefaultShape'>
    <Box DEF='DefaultBox' size='2 2 2'/>
    <Appearance>
     <Material diffuseColor='1 0.2 0.2'/>
    </Appearance>
   </Shape>
  </Transform>
  <Transform translation='-2.5 0 0'>
   <Shape>
    <Cone DEF='DefaultCone' bottom='true' bottomRadius='1' height='2' side='true'/>
    <Appearance>
     <Material diffuseColor='0.2 1 0.2'/>
    </Appearance>
   </Shape>
  </Transform>
  <Transform translation='0 0 0'>
   <Shape>
    <Cylinder DEF='DefaultCylinder' bottom='true' height='2' radius='1' side='true' top='true'/>
    <Appearance>
     <Material diffuseColor='0.2 0.2 1'/>
    </Appearance>
   </Shape>
  </Transform>
  <Transform translation='2.5 0 0'>
   <Shape>
    <Sphere DEF='DefaultSphere' radius='1'/>
    <Appearance>
     <Material diffuseColor='1 1 0.2'/>
    </Appearance>
   </Shape>
  </Transform>
  <Transform translation='4 0 0'>
   <Shape>
    <Text DEF='DefaultText' string='"hello" "X3D!"'>
     <FontStyle DEF='DefaultFontStyle'/>
    </Text>
    <Appearance DEF='DefaultAppearance'>
     <Material DEF='DefaultMaterial'/>
    </Appearance>
   </Shape>
  </Transform>
 </Scene>
```

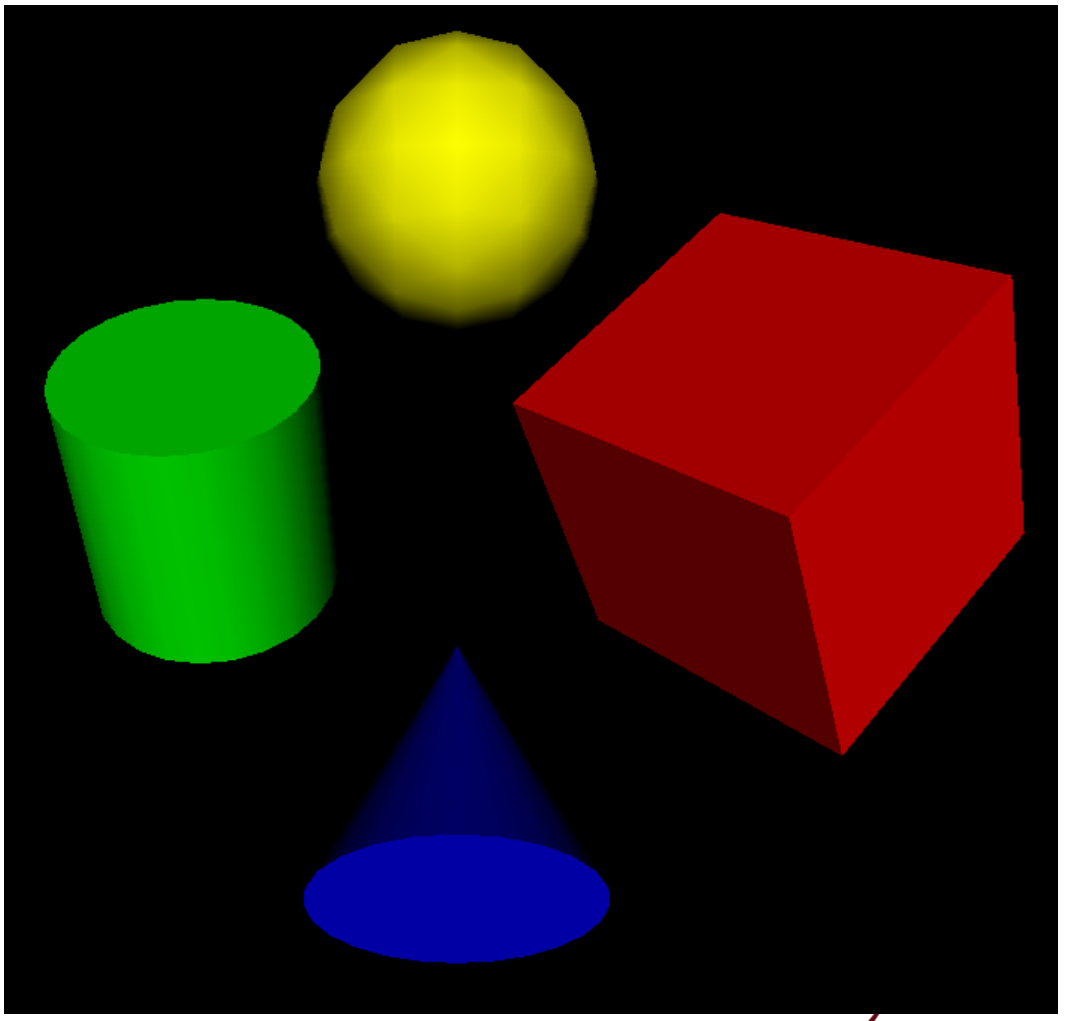


Transform nodes position each Shape so that they do not obscure each other

GeometryPrimitiveNodes.x3d

Transforming shapes



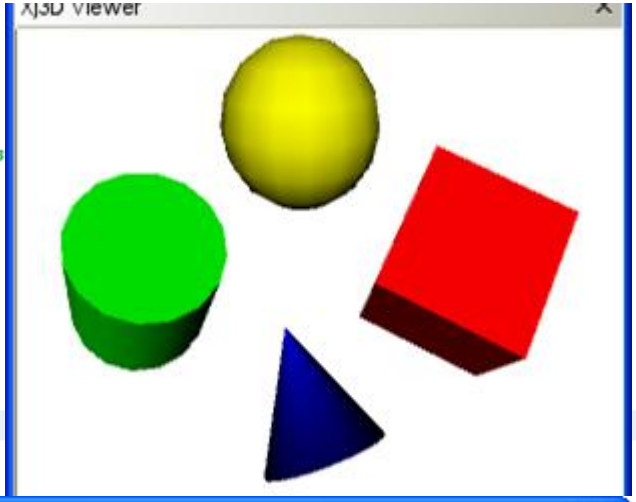


Transform.x3d

Transform.x3d - Editor

Transform.x3d

```
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter03-Grouping/Transform.x3d' name='identifier'/>
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <Background skyColor='1 1 1'/>
    <Viewpoint description='Book View' orientation='-0.682 -0.707 -0.187 0.68' pos
    <Transform rotation='1 1 1 1' translation='2 0 1'>
      <Shape>
        <Appearance>
          <Material diffuseColor='1 0 0'/>
        </Appearance>
        <Box/>
      </Shape>
    </Transform>
    <Transform translation='0 2 0'>
      <Shape>
        <Appearance>
          <Material diffuseColor='1 1 0'/>
        </Appearance>
        <Sphere/>
      </Shape>
    </Transform>
    <Transform rotation='1 0 0 .707' translation='-2 0 -1'>
      <Shape>
        <Appearance>
          <Material diffuseColor='0 1 0'/>
        </Appearance>
        <Cylinder/>
      </Shape>
    </Transform>
    <Transform rotation='1 0 0 -.707' translation='0 -2 0'>
      <Shape>
        <Appearance>
          <Material diffuseColor='0 0 1'/>
        </Appearance>
        <Cone/>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```



Edit Transform

DEF		containerField		
USE		children		
translation	0	2	0	
rotation	0	0	1	0
center	0	0	0	
scale	1	1	1	
scaleOrientation	0	0	1	0
bboxCenter	0	0	0	
bboxSize	-1	-1	-1	

OK Cancel Help

29:15 INS

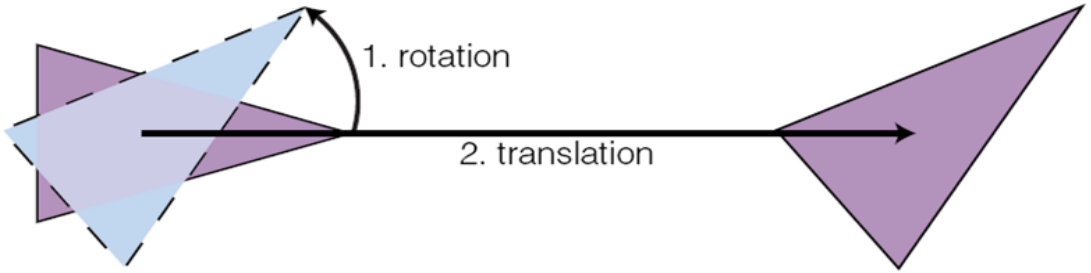
Order of transformation operations

The ordering of transformation operations is important and not symmetric. Algorithm:

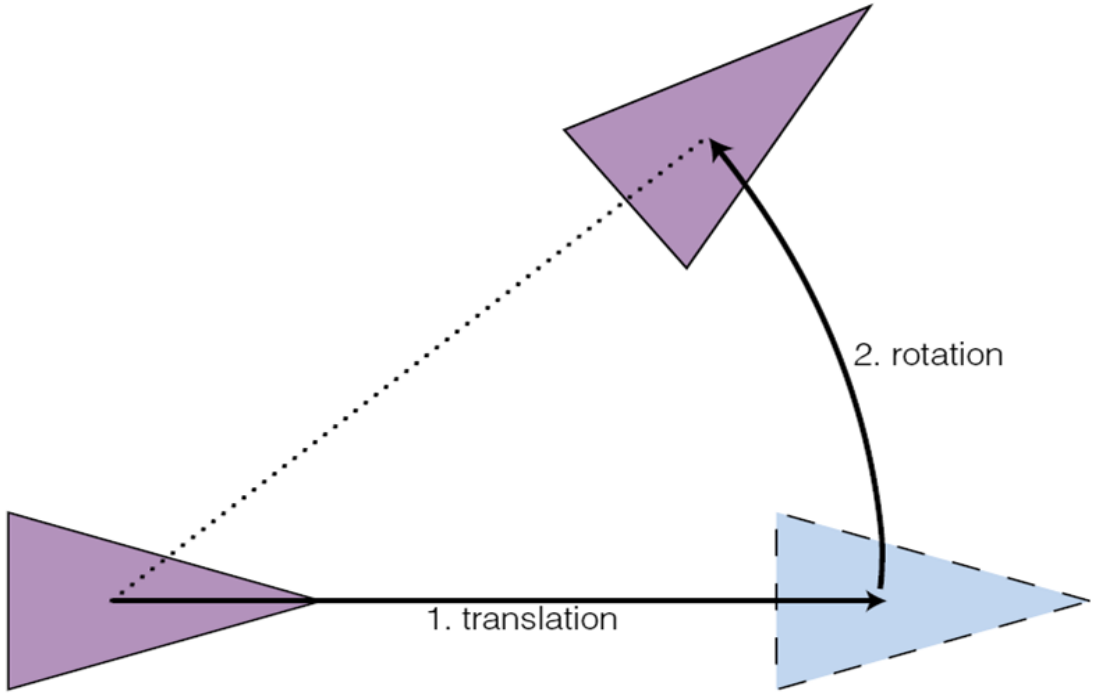
- Apply reverse *center* offset to set up for properly centered scaling and orientation operations
- Apply reverse *scaleOrientation*, then apply *scale* operation, then apply forward *scaleOrientation* to regain initial frame
- Apply *rotation* to final direction, then apply forward *center* offset to regain initial origin
- Apply *translation* to final location of new coordinate frame

Comparing out-of-order operations

Case 1



Case 2



Equivalent transformations

```
Transform {
        center C
        rotation R
        scale S
        scaleOrientation SR
        translation T
        children [...]
}
```

Using matrix transformation notation, where

- **C** (center),
- **SR** (scaleOrientation),
- **T** (translation),
- **R** (rotation), and
- **S** (scale)

are the equivalent transformation matrices, then

- **P'** is transformed child point **P**
- **P' = T · C · R · SR · S · −SR · −C · P**

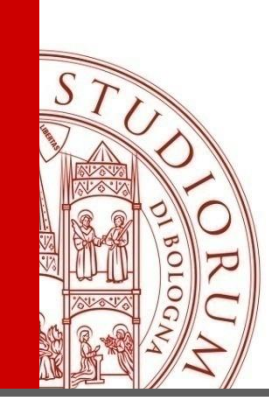
```
Transform {
        translation T
        children Transform {
                translation C
                children Transform {
                        rotation R
                        children Transform {
                                rotation SR
                                children Transform {
                                        scale S
                                        children Transform {
                                                rotation -SR
                                                children Transform {
                                                        translation -C
                                                        children [...]
                                                }
                                        }
                                }
                        }
                }
        }
}
```

Suggested Exercise 2

With X3D-Edit

- Create a new X3D scene, Save As using a new filename of your choosing
- Iconize the <head> element by clicking margin '+'
- Drag and drop nodes to build the scene
  - Create a simple object using only primitive geometric shapes
- Edit by typing, and by using node editors
- Make sure you maintain valid XML as you go
- Save, view, repeat as necessary
- **Right-click to launch external viewer**

This matches how we build many X3D scenes



Geometry:

1. Primitive Shapes
2. **Points, Lines and Polygons**
3. Geometry2D Nodes
4. Triangles and Quadrilaterals
5. NURBS

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

Indexed**Line**Set 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*

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/polyline

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

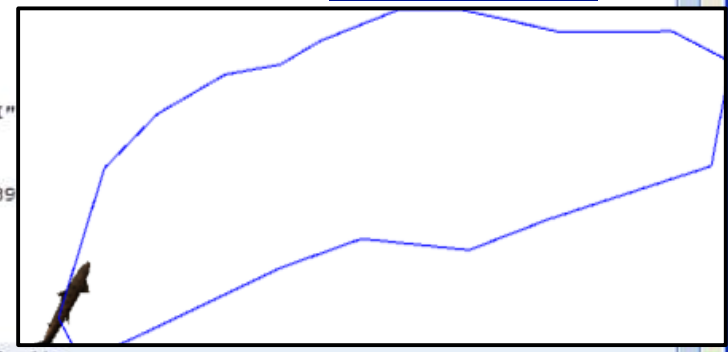
DEF TurnPoints

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

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
- *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

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-inst
4    <head>
5      <meta content='Color.x3d' name='title'/>
6      <meta content='Vertex color applied to IndexedFaceSet built positive-displacement c
7                    static because the PositionInterpolators contain an initial offest c
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-GeometryP
14     <meta content='KelpTank.x3d' name='reference'/>
15     <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/KelpForestExhibit/P
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
22     <ExternProtoDeclare name='WhereAmI' url='"../Chapter14-Prototypes/WhereAmI.x3d#Wher
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
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 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
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 0.625 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>
```

Edit Coordinate

DEF CoordinateNodeExample

USE

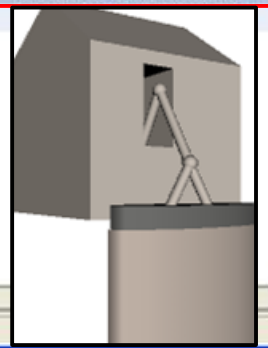
containerField coord

point array

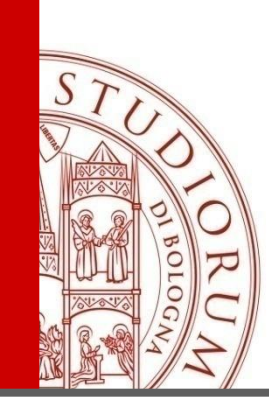
	x	y	z
0	0	0	0
1	2	0	0
2	2	1.75	0
3	0	1.75	0
4	0.625	0.75	0
5	1	0.75	0
6	1	1.6	0
7	0.625	1.6	0
8	0.625	0.75	-0.65
9	1	0.75	-0.65
10	1	1.6	-0.65
11	0.625	1.6	-0.65
12	2	0	-2.7
13	2	1.75	-2.7
14	2	2.5	-1
15	0	0	-2.7
16	0	1.75	-2.7
17	0	2.5	-1

+ -

OK Cancel Help



37:1 INS



Geometry:

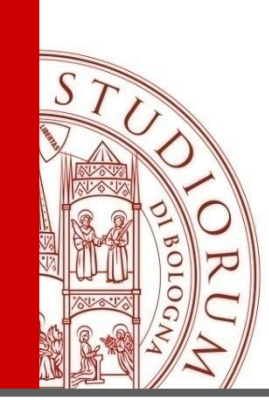
1. Primitive Shapes
2. Points, Lines and Polygons
3. **Geometry2D Nodes**
4. Triangles and Quadrilaterals
5. NURBS

Overview: Geometry2D Nodes

These are simple utility (convenience) nodes

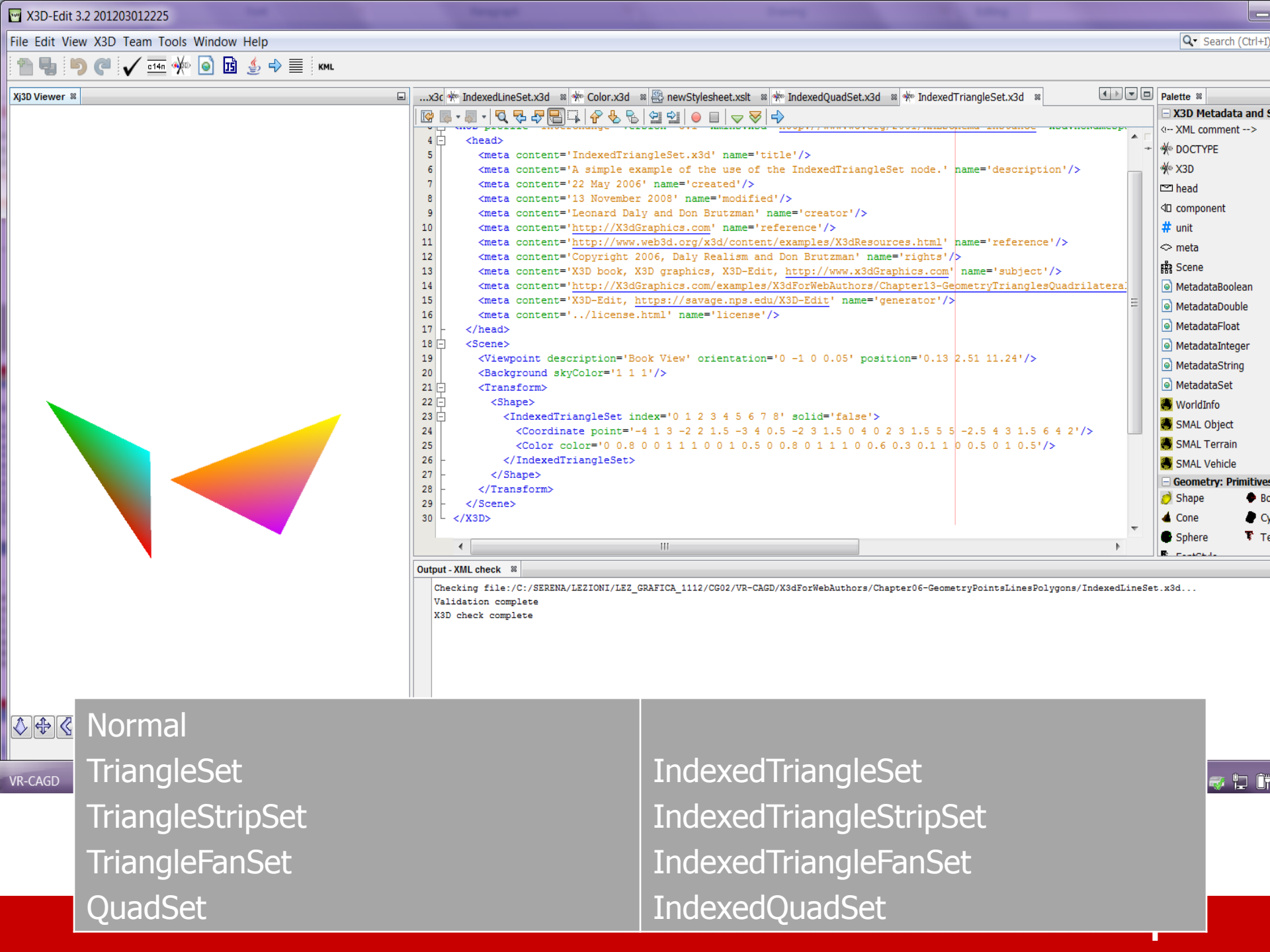
Geometry2D nodes

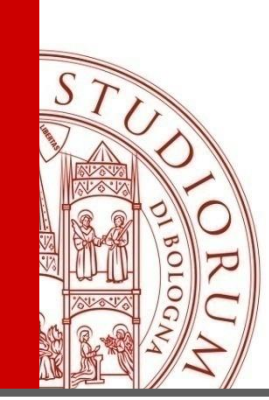
- Arc2D lines
- ArcClose2D polygonal shape
- Circle2D lines
- Disk2D polygonal shape
- Polyline2D lines
- Polypoint2D points
- Rectangle2D polygonal shape
- TriangleSet2D polygonal shapes



Geometry:

1. Primitive Shapes
2. Points, Lines and Polygons
3. Geometry2D Nodes
4. **Triangles and Quadrilaterals**
5. NURBS


X3D-Edit 3.2 201203012225
File Edit View X3D Team Tools Window Help
Search (Ctrl+I)
Xj3D Viewer
...x3d
IndexedLineSet.x3d
Color.x3d
newStylesheet.xslt
IndexedQuadSet.x3d
IndexedTriangleSet.x3d
Palette
X3D Metadata and S
XML comment -->
DOCTYPE
X3D
head
component
unit
meta
Scene
MetadataBoolean
MetadataDouble
MetadataFloat
MetadataInteger
MetadataString
MetadataSet
WorldInfo
SMAL Object
SMAL Terrain
SMAL Vehicle
Geometry: Primitives
Shape
Cone
Sphere
4 <head>
5 <meta content='IndexedTriangleSet.x3d' name='title'/>
6 <meta content='A simple example of the use of the IndexedTriangleSet node.' name='description'/>
7 <meta content='22 May 2006' name='created'/>
8 <meta content='13 November 2008' name='modified'/>
9 <meta content='Leonard Daly and Don Brutzman' name='creator'/>
10 <meta content='http://X3dGraphics.com' name='reference'/>
11 <meta content='http://www.web3d.org/x3d/content/examples/X3dResources.html' name='reference'/>
12 <meta content='Copyright 2006, Daly Realism and Don Brutzman' name='rights'/>
13 <meta content='X3D book, X3D graphics, X3D-Edit, http://www.x3dGraphics.com' name='subject'/>
14 <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter13-GeometryTrianglesQuadrilatera
15 <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
16 <meta content='../license.html' name='license'/>
17 </head>
18 <Scene>
19 <Viewpoint description='Book View' orientation='0 -1 0 0.05' position='0.13 2.51 11.24'/>
20 <Background skyColor='1 1 1'/>
21 <Transform>
22 <Shape>
23 <IndexedTriangleSet index='0 1 2 3 4 5 6 7 8' solid='false'>
24 <Coordinate point='-4 1 3 -2 2 1.5 -3 4 0.5 -2 3 1.5 0 4 0 2 3 1.5 5 5 -2.5 4 3 1.5 6 4 2'/>
25 <Color color='0 0.8 0 0 1 1 1 0 0 1 0.5 0 0.8 0 1 1 1 0 0.6 0.3 0.1 1 0 0.5 0 1 0.5'/>
26 </IndexedTriangleSet>
27 </Shape>
28 </Transform>
29 </Scene>
30 </X3D>
Output - XML check
Checking file:/C:/SERENA/LEZIONI/LEZ_GRAFICA_1112/CG02/VR-CAGD/X3dForWebAuthors/Chapter06-GeometryPointsLinesPolygons/IndexedLineSet.x3d...
Validation complete
X3D check complete
VR-CAGD
Normal
TriangleSet
TriangleStripSet
TriangleFanSet
QuadSet
IndexedTriangleSet
IndexedTriangleStripSet
IndexedTriangleFanSet
IndexedQuadSet



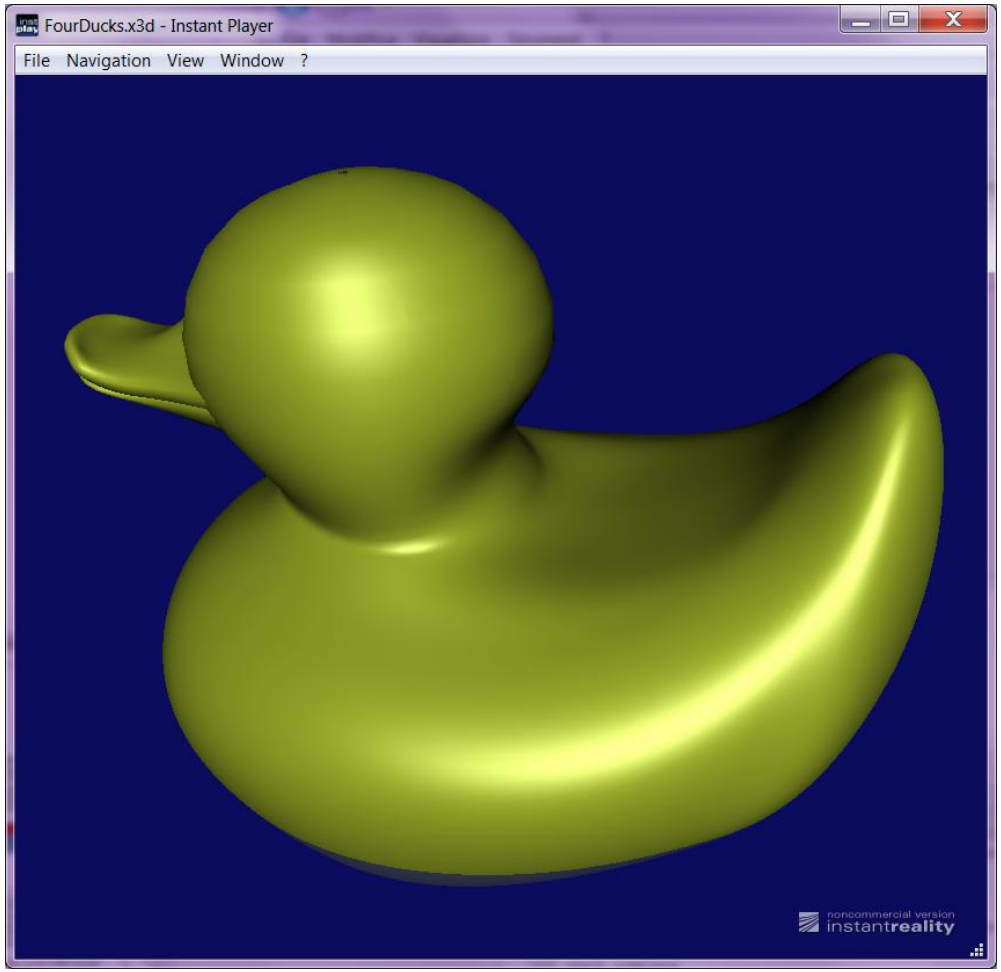
Geometry:

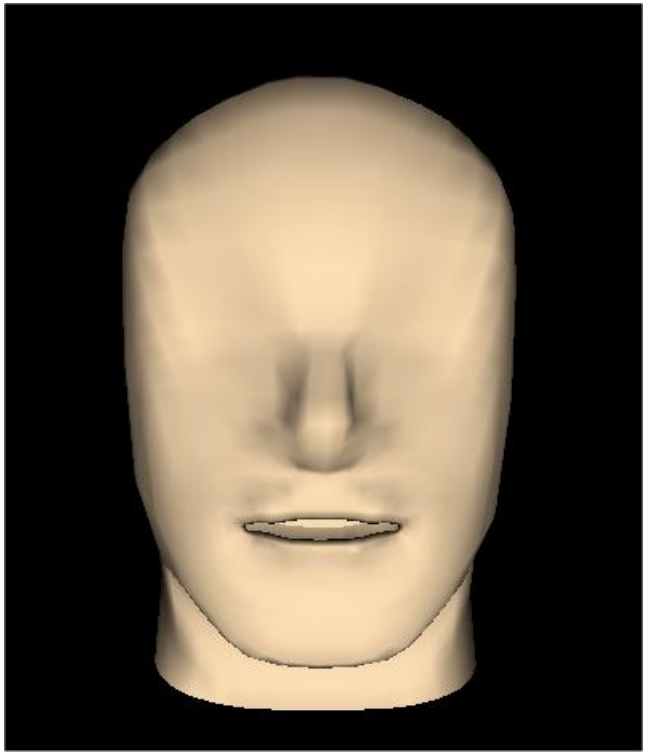
1. Primitive Shapes
2. Points, Lines and Polygons
3. Geometry2D Nodes
4. Triangles and Quadrilaterals
5. **NURBS**

NURBS nodes

Non-uniform Rational B-Spline (NURBS) nodes define parametric surfaces

- Precise, accurate, terse, scalable representations since mathematically defined
- Can be tessellated as high-fidelity polygonal surface at a resolution appropriate to viewer distance
- Difficult to author without special tools
- X3D NURBS nodes include: Contour2D, ContourPolyline2D, CoordinateDouble, NurbsCurve, NurbsCurve2D, NurbsOrientationInterpolator, NurbsPatchSurface, NurbsPositionInterpolator, NurbsSet, NurbsSurfaceInterpolator, NurbsSweptSurface, NurbsSwungSurface, NurbsTextureCoordinate, NurbsTrimmedSurface





Head: this surface is created by 24 points in the *u direction and 30 points in the v direction for a total of 720* control points.

web|3D CONSORTIUM

< X3D >

Construction techniques

A) special cases of NURBS surfaces such as sphere, cylinder or Bézier surfaces;

B) Extrusion/swept surfaces, constructed given a spine curve and a cross-section curve either or both of which can be NURBS curves;

C) surfaces of revolution, constructed given a circle/arc and a NURBS cross-section curve;

D) skinned surfaces constructed from a set of curves;

E) Gordon surfaces interpolating two sets of curves;

F) Coons patches, a bi-cubic blended surface constructed from four border curves;

G) Surfaces interpolating a set of points.

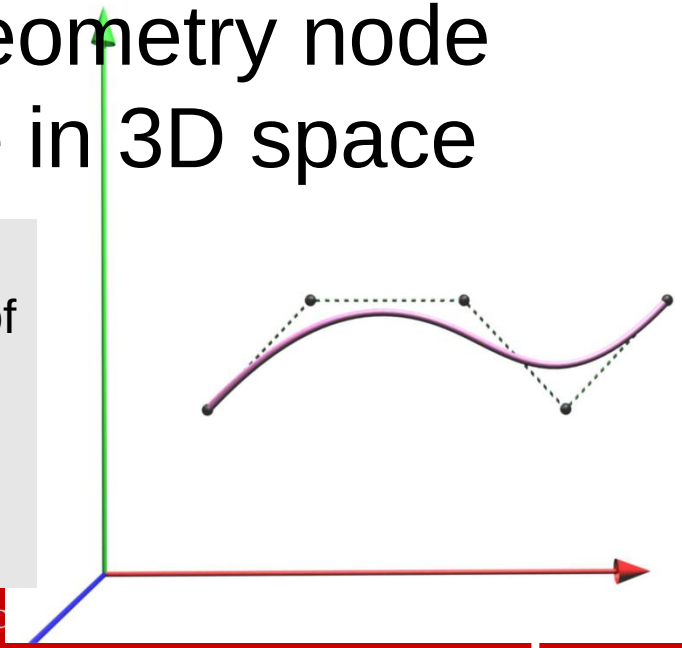
NurbsCurve

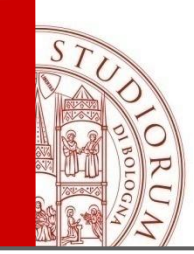
```
NurbsCurve : X3DParametricGeometryNode {
    SFNode      [in,out] controlPoint   []       [X3DCoordinateNode]
    SFNode      [in,out] metadata       NULL     [X3DMetadataObject]
    SFInt32     [in,out] tessellation   0 (-&,&)
    MFDouble    [in,out] weight         [] (0,&)
    SFBoolean   []       closed         FALSE
    MFDouble    []       knot           [] (-&,&)
    SFInt32     []       order          3 [2,&)
}
```

The NurbsCurve node is a geometry node defining a parametric curve in 3D space

knots defines the knot vector.

- The number of knots shall be equal to the number of control points plus the order of the curve.
- The order shall be non-decreasing.
- Within the knot vector there may not be more than order−1 consecutive knots of equal value.





Control points are in homogeneous coordinates

- The control point is actually a 4D vector (x, y, z, weight), which means that it's actual 3D position is (x/weight, y/weight, z/weight). Instead of:

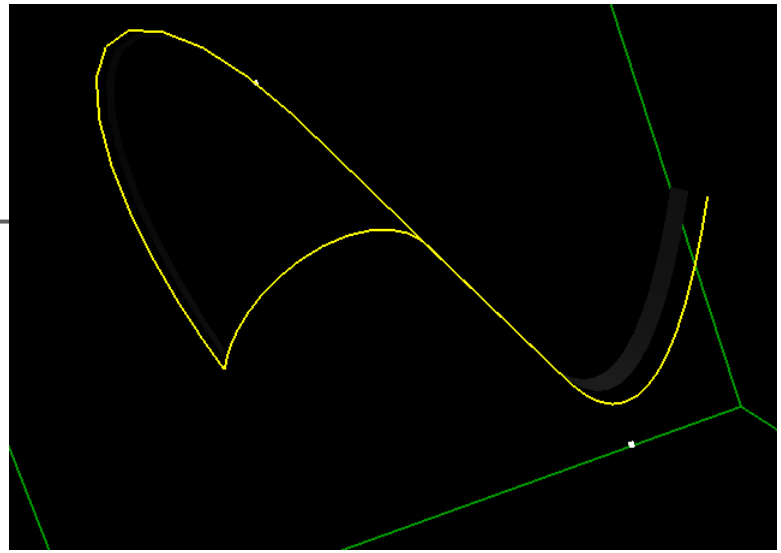
```
P(u) = (sum of basis * control point * weight)/(sum of basis * weight)
```

  X3D uses a simpler equation:

```
P(u) = (sum of basis * control point) / (sum of basis * weight)
```

That is, *"X3D control point"* (as specified in X3D file) is assumed to be already multiplied by weight.

- If you want to intuitively pull the curve toward the control point, you should
  - Calculate *"normal control point"* (3D, not in homogeneous coordinates) as *"X3D control point / weight"*.
  - Increase the weight (to pull the curve toward *"normal control point"*), or decrease (to push the curve away from it).
  - Calculate new *"X3D control point"* as *"normal control point * new weight"*.
- In other words, if you just want to increase the weight 2 times, then the corresponding control point should also be multiplied * 2, to make things behave intuitive.



```
<Shape>
        <Appearance>
                <Material emissiveColor="1 1 0" />
        </Appearance>
        <NurbsCurve         tessellation="40"
                weight="1 0.25 1 1 1">
                <Coordinate containerField="controlPoint"
                point="-5 0 2 -2.5 5 2 0 0 2 2.5 -5 2 5 0 2" />
        </NurbsCurve>
</Shape>
<Shape>
        <Appearance>
                <Material emissiveColor="1 1 0" />
        </Appearance>
        <NurbsCurve         tessellation="40"
                weight="1 0.25 1 1 1">
                <Coordinate containerField="controlPoint"
                point="-5 0 2 -0.625 1.25 0.5 0 0 2 2.5 -5 2 5 0 2" />
        </NurbsCurve>
</Shape>
```

A closed Spline curve can be specified by repeating the limiting control points, specifying a periodic knot vector, and setting the closed field to TRUE.

NurbsCurve2D

```
NurbsCurve2D : X3DNurbsControlCurveNode {
    MFVec2d      [in,out]          controlPoint     []       (-&,&)
    SFNode       [in,out]          metadata         NULL [X3DMetadataObject]
    SFInt32      [in,out]          tessellation     0        (-&,&)
    MFDouble     [in,out]          weight           []       (0,&)
    SFBool       []                closed           FALSE
    MFDouble     []                knot             []       (-&,&)
    SFInt32      []                order            3        [2,&)
}
```

X3DNurbsControlCurveNode

```
X3DNurbsControlCurveNode : X3DNode {
   MFVec2d [in,out] controlPoint [] (-&,&)
   SFNode [in,out] metadata NULL [X3DMetadataObject]
}
```

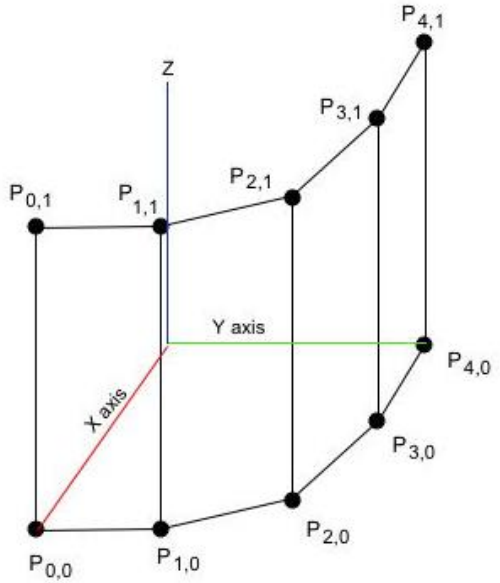
The control points are defined in 2D coordinate space and interpreted according to the descendent node type as well as the user of this node instance.

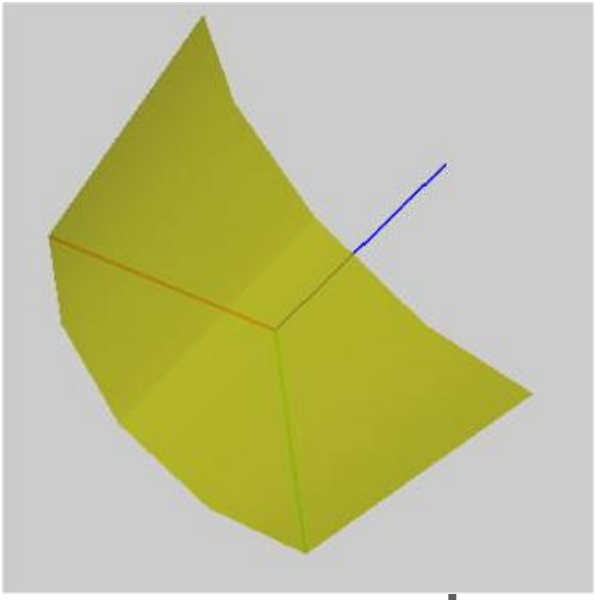
X3DNurbsSurfaceGeometryNode

```
X3DNurbsSurfaceGeometryNode : X3DParametricGeometryNode {
SFNode   [in,out] controlPoint     []       [X3DCoordinateNode]
SFNode   [in,out] metadata         NULL     [X3DMetadataObject]
SFNode   [in,out] texCoord         []
         [X3DTextureCoordinateNode|NurbsTextureCoordinate]
SFInt32  [in,out] uTessellation    0 (-&,&) % if 0, the number of tessellation points is:
SFInt32  [in,out] vTessellation    0 (-&,&) %          (2 ×(u/v)dimension)+1
MFDouble [in,out] weight           []       (0,&)
SFBool   []       solid            TRUE     %visible when viewed from the inside
SFBool   []       uClosed          FALSE
SFInt32  []       uDimension       0        [0,&)     % number of CP in u.
MFDouble []       uKnot            []       (-&,&)
SFInt32  []       uOrder           3        [2,&)
SFBool   []       vClosed          FALSE
SFInt32  []       vDimension       0        [0,&)      % number of CP in v.
MFDouble []       vKnot            []       (-&,&)
SFInt32  []       vOrder           3        [2,&)
}
```

The X3DNurbsSurfaceGeometryNode represents a geometry node defining a parametric surface for all types of NURBS surfaces

- The geometry of the patch is specified by an array of control points with N_R rows and N_C columns, orders D_R , D_C
- The definition of the spline functions requires that the spline parameters and dimensions of the control points array satisfy:
- $K_R = N_R + D_R$ $K_C = N_C + D_C$
- The weights and control point arrays are flattened into lists by traversing the arrays in column major order.
- controlPoint ← $P_{0,0}$, $P_{1,0}$, $P_{2,0}$, ...$P_{4,0}$,$P_{0,1}$,...,$P_{4,1}$;
  each P a SFVec3f value.
- weight ← $w_{0,0}$, $w_{1,0}$, $w_{2,0}$, ...$w_{4,0}$,$w_{0,1}$,...,$w_{4,1}$;
  each w a SFFloat value.
- uKnot ← row-parameter knot vector, of length K_R=8
- uDimension ← N_R=5
- uOrder ← D_R=3
- vKnot ← column-parameter knot vector, of length K_C=4
- vDimension ← N_C=2
- vOrder ← D_C=2





The control vertex corresponding to the control point P[i,j] on the control grid is in column major order :

$P[i,j].x = controlPoint[i + (j \times uDimension)].x$

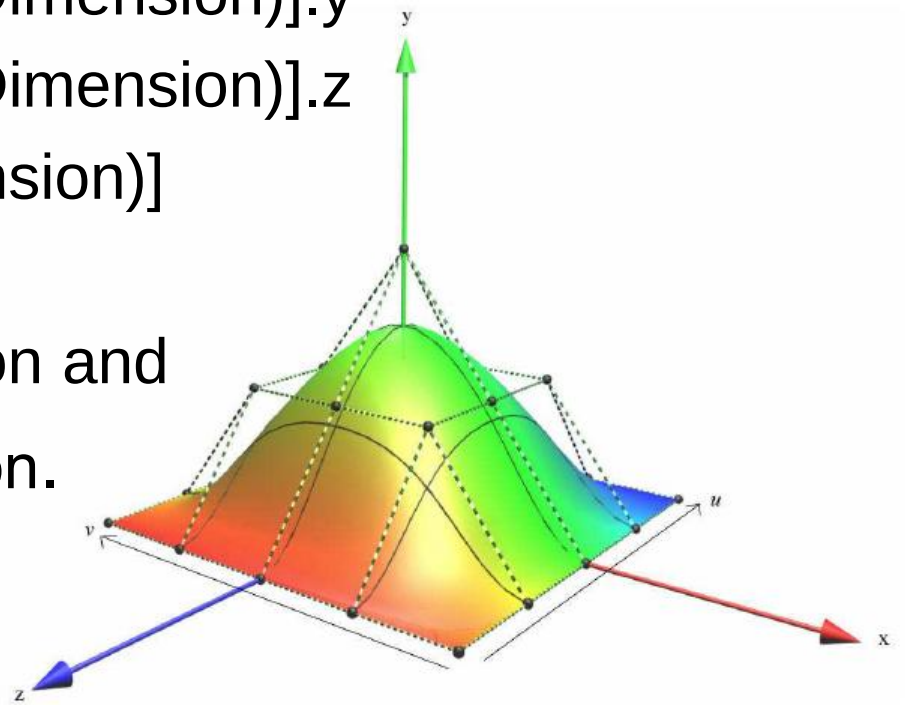
$P[i,j].y = controlPoint[i + (j \times uDimension)].y$

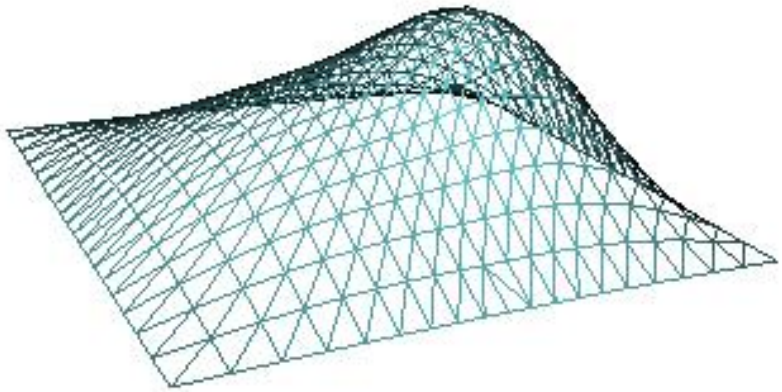
$P[i,j].z = controlPoint[i + (j \times uDimension)].z$

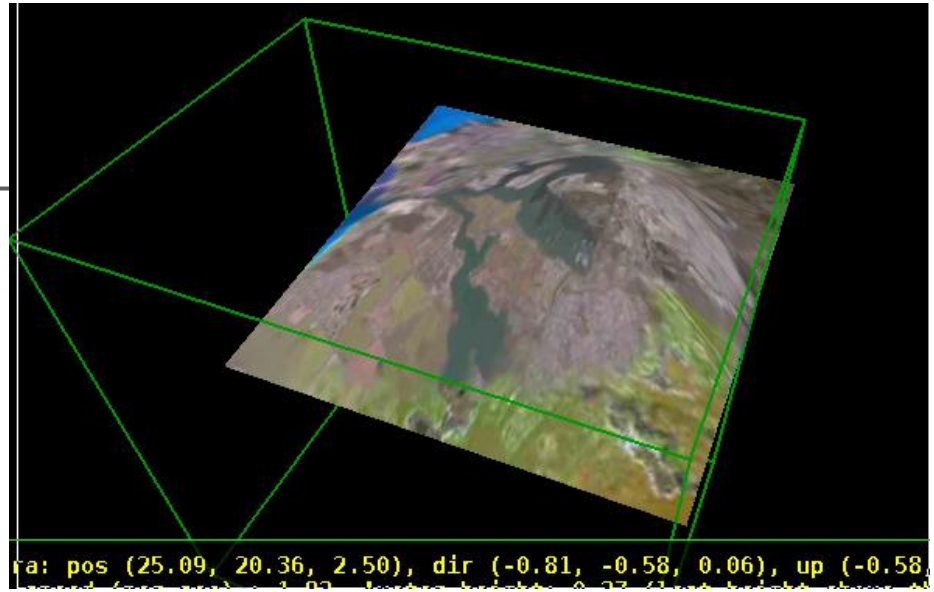
$P[i,j].w = weight[i + (j \times uDimension)]$

where $0 \leq i < uDimension$ and

$0 \leq j < vDimension$.








```
<Shape>
    <NurbsPatchSurface DEF='NS' solid='false' uDimension='5' uOrder='4' uTessellation='30' vDimension='5'
    vOrder='4' vTessellation='30'>
      <Coordinate containerField='controlPoint' point='-10 -10 0 -10 -5 0 -10 0 0 -10 5 0 -10 10 0 -5 -10 0 -5 -5 2.5 -5 0
    5 -5 5 2.5 -5 10 0 0 -10 0 0 -5 2.5 0 0 5 0 5 2.5 0 10 0 5 -10 0 5 -5 2.5 5 0 15 5 5 2.5 5 10 0 10 -10 0 10 -5 0 10 0 0
    10 5 0 10 10 0'/>
    </NurbsPatchSurface>
    <Appearance>
      <ImageTexture url='"PearlHarborLowResolution.jpg“ />
    </Appearance>
</Shape>
```

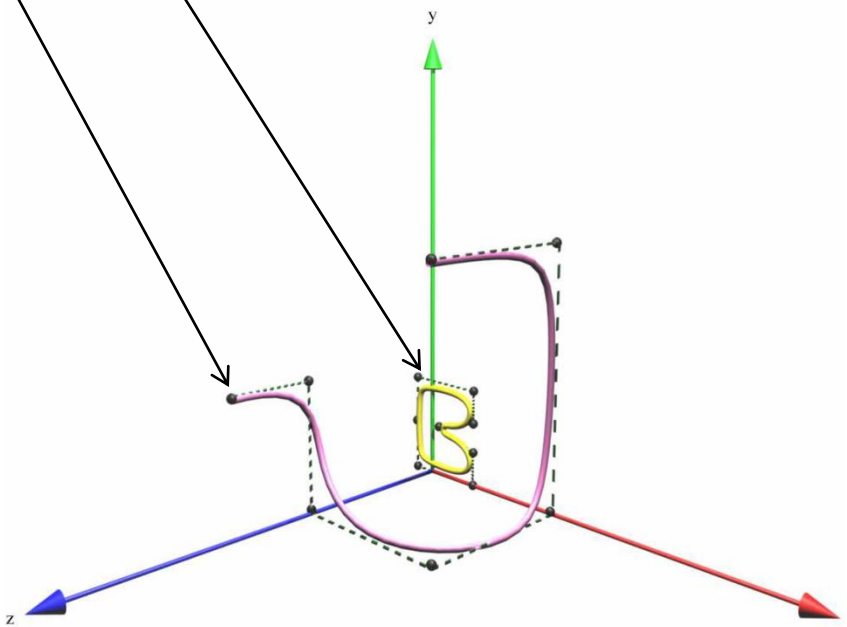
The higher the tessellation, the smoother the surf will appear on screen, however the more computationally expensive the surf becomes.

NurbsSweptSurface

```
NurbsSweptSurface : X3DParametricGeometryNode {
   SFNode [in,out] crossSectionCurve  [] X3DNurbsControlCurveNode]
   SFNode [in,out] metadata           NULL [X3DMetadataObject]
   SFNode [in,out] trajectoryCurve    [] [NurbsCurve]
   SFBool []     ccw                  TRUE
   SFBool []     solid                TRUE
}
```

Conceptually it is the NURBS equivalent of the Extrusion

To have the polygons' normals facing away from the axis, the trajectory curve should be oriented so that it is moving counterclockwise when looking down the −Y axis, thus defining a concept of "inside" and "outside".



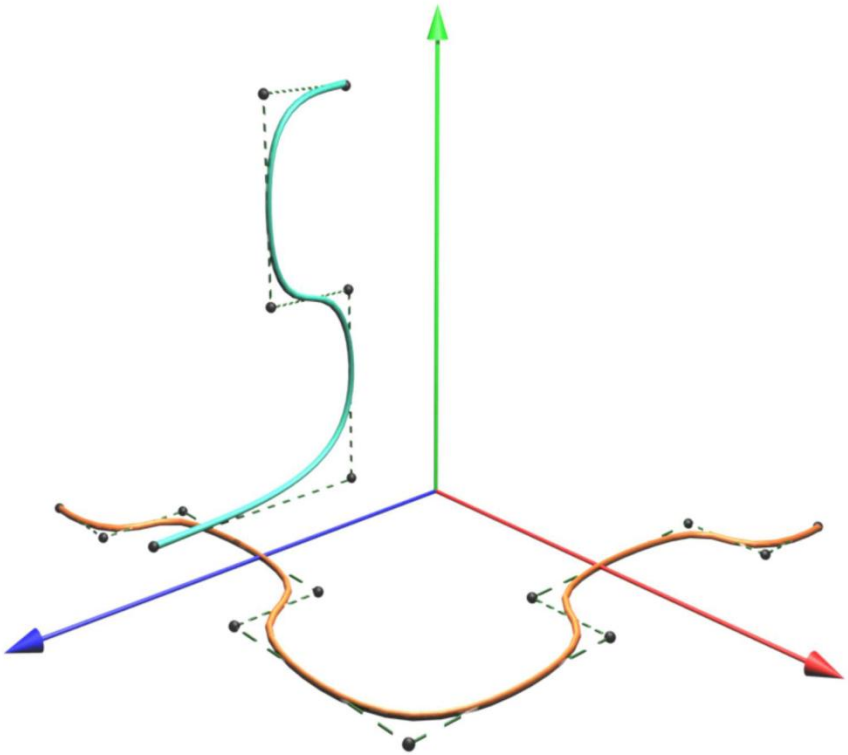
NurbsSwungSurface

```
NurbsSwungSurface : X3DParametricGeometryNode {
    SFNode [in,out]     metadata          NULL [X3DMetadataObject]
    SFNode [in,out]     profileCurve      [] [X3DNurbsControlCurveNode]
    SFNode [in,out]     trajectoryCurve   [] [X3DNurbsControlCurveNode]
    SFBool []           ccw TRUE
    SFBool []           solid TRUE
}
```

Defines a path and constant cross section of the path

The **profile curve** is a 2D curve in the yz-plane that describes the cross-sectional shape of the object.

The **trajectory curve** is a 2D curve in the xz-plane that describes the path over which to trace the cross-section.



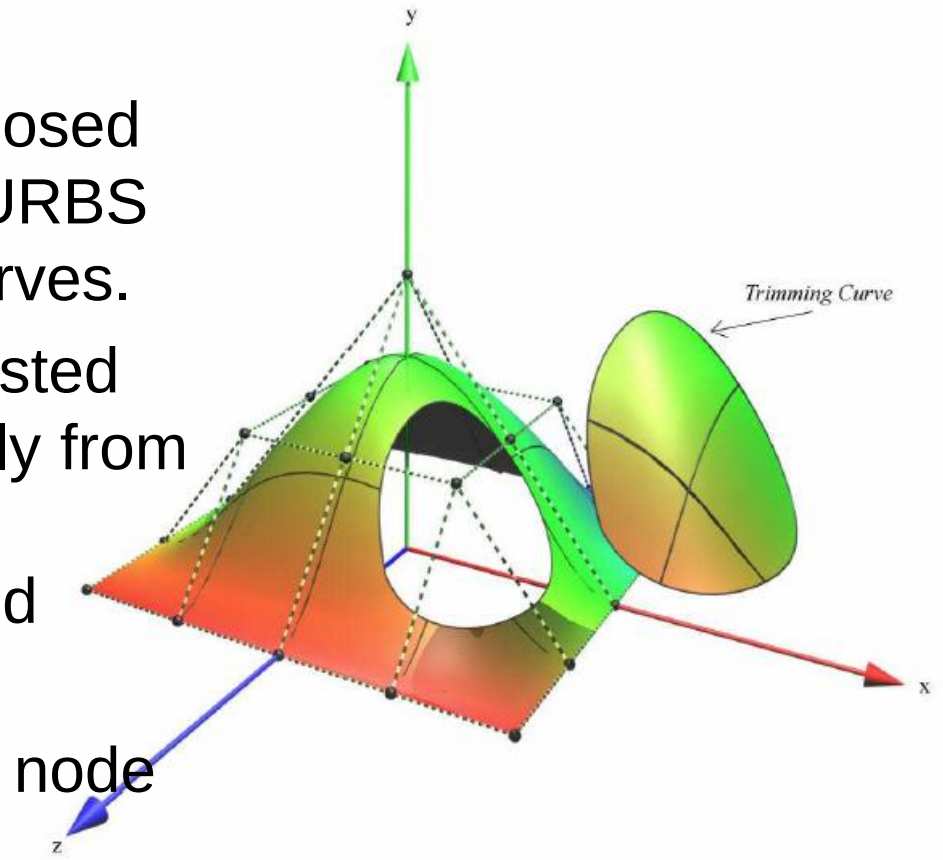
NurbsTrimmedSurface

```
X3DNurbsSurfaceGeometryNode {
      MFNode          [in]      addTrimmingContour              [Contour2D]
      MFNode          [in]      removeTrimmingContour           [Contour2D]
      SFNode          [in,out]  controlPoint                    [] [X3DCoordinateNode]
      SFNode          [in,out]  metadata                        NULL [X3DMetadataObject]
      SFNode          [in,out]  texCoord [] [X3DTextureCoordinateNode|NurbsTextureCoordinate]
      MFNode          [in,out]  trimmingContour                 [] [Contour2D]
      SFInt32         [in,out]  uTessellation                   0 (-&,&)
      SFInt32         [in,out]  vTessellation                   0 (-&,&)
      MFDouble        [in,out]  weight                          [] (0,&)
      SFBool          []        solid                           TRUE
      SFBool          []        uClosed                         FALSE
      SFInt32         []        uDimension                      0 [0,&)
      MFDouble        []        uKnot                           [] (-&,&)
      SFInt32         []        uOrder                          3 [2,&)
      SFBool          []        vClosed                         FALSE
      SFInt32         []        vDimension                      0 [0,&)
      MFDouble        []        vKnot                           [] (-&,&)
      SFInt32         []        vOrder                          3 [2,&)
}
```

The trimming curve specifies a NURBS-curve that limits the NURBS surface in order to create NURBS surfaces that contain holes or have smooth boundaries. Trimming curves are curves in the parametric space of the surface..

NurbsTrimmedSurface

- A trimming region is defined by a set of closed trimming loops in the parameter space of a surface.
- A trimming loop consists of a closed and connected sequence of NURBS curves and piecewise linear curves.
- Loops may be nested, but a nested loop must be oriented oppositely from the loop that contains it. The outermost loop must be oriented counter-clockwise.
- Trimming loops are Contour2D node



Contour2D

```
Contour2D : X3DNode {
    MFNode [in]          addChildren      [NurbsCurve2D|ContourPolyline2D]
    MFNode [in]          removeChildren   [NurbsCurve2D|ContourPolyline2D]
    MFNode [in,out]      children         [] [NurbsCurve2D|ContourPolyline2D]
    SFNode [in,out]      metadata         NULL [X3DMetadataObject]
}
```

The Contour2D node groups a set of curve segments to a composite contour.

The children shall form a closed loop with the first point of the first child repeated as the last point of the last child and the last point of a segment repeated as the first point of the consecutive one.

Example

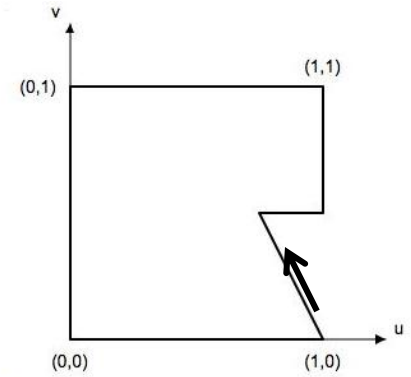


Figure 8: trimming curve

controlPoint=' 0.0000e+00 0.0000e+00,
1.0000e+00 0.0000e+00,
7.5000e-01 5.0000e-01,
1.0000e+00 5.0000e-01,
1.0000e+00 1.0000e+00,
0.0000e+00 1.0000e+00,
0.0000e+00 0.0000e+00‘

Since this loop is traversed counter-clockwise it would be expected that the concave side of the surface should be visible

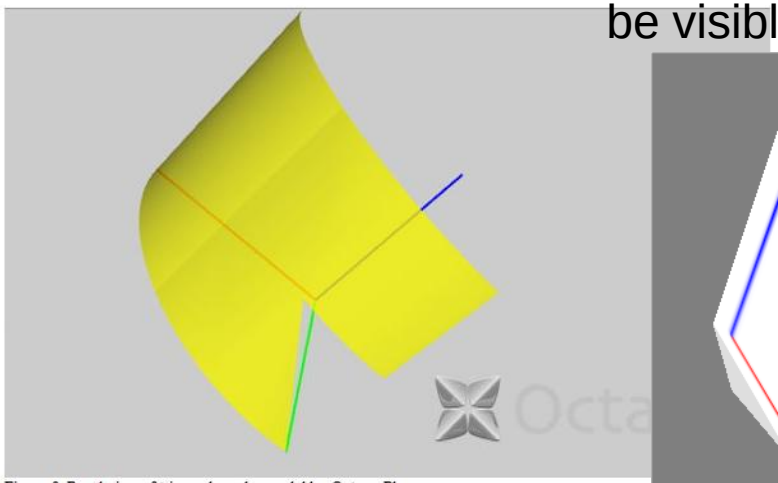
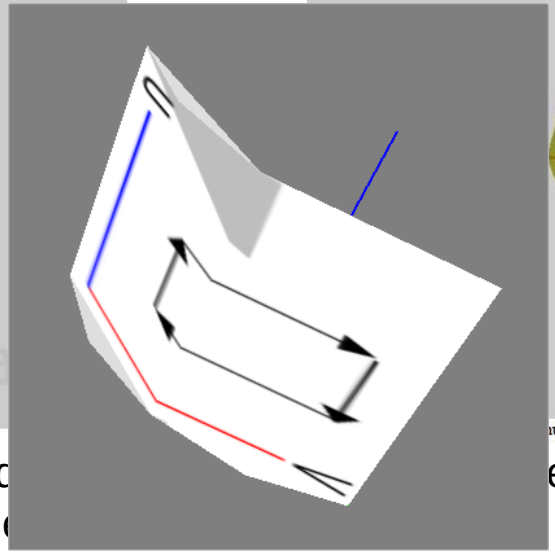
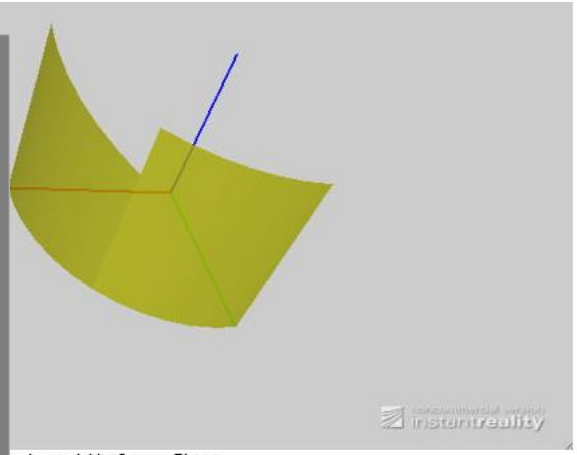


Figure 9: Rendering of trimmed_nurbs model by Octaga Player.



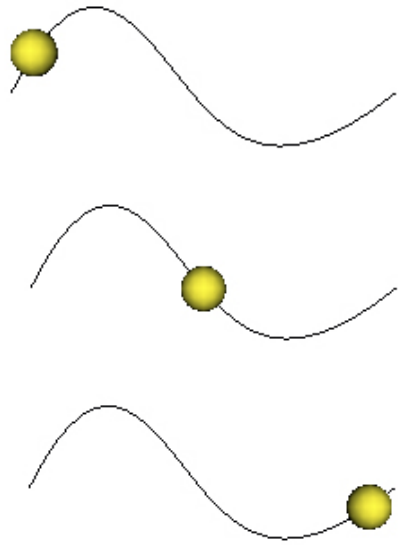


urbs model by Instant Player.

Instant Player browser has made a c ed patch coordinates:
v •row-parameter , u column-parame

Nurbs Position Interpolator

```
NurbsPositionInterpolator : X3DInterpolatorNode {
   SFFloat      [in]      set_fraction (-∞ ,∞ )
   SFBool       [in,out]  fractionAbsolute TRUE
   MFDouble     [in,out]  key [](-∞ ,∞ )
   MFVec3f      [in,out]  keyValue [](-∞ ,∞ )
   MFDouble     [in,out]  weight [] (-∞ ,∞ )
   MFDouble     [in,out]  knot [] (-∞ ,∞ )
   SFInt32      [in,out]  order 3 (2,∞ )
   SFVec3f      [out]     value_changed
}
```



The true power of a NurbsCurve is the ability to animate an object along the curve as an animation path.

However, there are no control points specified. Instead, control vertices are found in the keyValue field. The “key” field defines time points at which the value in the keyValue field will be reached.

The only control points (keyValues) that the path is guaranteed to touch are the first and last.

Inline node

Loads another X3D world within current scene

- Supported formats depend on user's X3D browser
- XML .x3d, ClassicVRML .x3dv,
- Compressed binary .x3db, possibly VRML97 .wrl

Inline scene is positioned, rotated and scaled to match the local coordinate frame

- Local reference frame determined by parent Transformation node hierarchy
- User's viewpoint does not change automatically to the loaded Inline scene's default Viewpoint

Suggested Exercise 3

- Use existing model from another tool (e.g. Blender)
- Save as in XML as .x3d file
- Load (or import) into X3D-Edit, fix bugs (if any)
- Create parent scene that loads first via Inline
- Add further X3D content to parent scene
  - Create a simple object using only NURBS surface shapes