CS297 Report

3D Web Graphics without Plugins using VML

Jiewei (Joyce) Lin (lin_jiewei@yahoo.com)

Advisor: Dr. Chris Pollett

1. Introduction

At the present time there is no way to directly view 3D graphics in the most common browsers (Internet Explorer and Netscape) without using a plugin. The closest thing that exists is LiveGraphics3D [MK01], a Java 1.1 applet for viewing Mathematica objects and a weak XML language that is translated using Javascript to VML in Internet Explorer [GR00]. VML (Vector Markup Language) [W3C98b] is an XML [W3C98] based mark-up language for vector graphics. It is natively supported in Internet Explorer 5 and above, the most commonly used browsers on the web today. The goal of this project is to develop a stylesheet-transformation from the X3D language [W3DC01] to VML.

My project will provide an Internet Explorer solution to plugin-less viewing of 3D graphics. Paungkaew Sangtrakulcharoen also at SJSU is working on a similar project but with target language SVG [W3C01] which is a stronger vector mark-up language. Her project will provide a Netscape solution to plugin-less viewing of 3D graphics. X3D is a version of VRML (virtual reality modeling language) [ANM97] specified as an XML DTD. It is a W3C standard and supports a robust set of tags for describing 3D objects and their behaviors. A stylesheet transformation from X3D to VML would, thus, allow web-sites to exploit 3D without having to worry about whether the client user is patient enough and competent enough to download and install a plugin.

To understand why this project is feasible and yet also rises to the level of a master's project it is important to understand a little about the underlying technologies involved. 3D objects are typically modeled as a mesh of 2D-polygons such as triangles. These polygons are typically shaded so that the surface of the object appears smooth and the underlying polygons are not noticeable. To draw the 3D object on a computer screen in a particular orientation a projection and clipping operation is applied to the object to get a 2D view. VML supports drawing of 2D-shapes such as polygons at given locations on the screen (although it has limited support for affine transformations of these shapes). It also supports gradient painting of shapes. Thus, it is conceivable using these tools to render 3D-objects. To take an X3D file and produce an VML image of a particular view of the object it represents involves generating a mesh corresponding to the X3D object, calculating shading and doing the final 2D projection. This involves implementing several non-trivial graphics algorithms in our translator. In many 3D situations the client will want to view the 3D objects from different perspectives, so we also need to be able to recompute the image on the client side in these perspectives. On the other hand, we need to optimize both the size and speed of our translator so that clients will not

become bored by the download time of the translator and the time of the translation. As we describe below, the technology which is available to do this translation is quite limited which partly explains why it has not been done before.

The translator for this project will be written using style-sheet transformations. Basically these are rules which are applied by the browser or by the server to a tag when it is read or before it is transmitted. Internet Explorer supports the stylesheet transformation language XSLT [W3C99]. XSLT supports tag-replacement by other tags as well as manipulating tag-attributes. However, as we need to be able to manipulate matrices will probably have to tie these transformations to ECMAscript (aka Javascript) code [ECMA99] which can in theory output VML code. Further, the style transformations and this local ECMAscript code is typically only processed once so care must be taken to leave hooks so that different orientations of the object can still be computed. These considerations as well as speed considerations make the job of writing this translator much harder than the writing of a standard renderer and further illustrate that this project rises to the level of a Master's project.

This document is organized as follows. Section 2 introduces an image of a sunset in VML. Section 3 explores a 3D mock-up apartment in VRML. Section 4 shows histograms translated to VML. Section 5 contains a box translated from X3D to VML. Section 6 lists challenges for this project. Section 7 summarizes what I have done so far and gives a brief overview of my future work.

2. Deliverable 1: An image of a sunset in VML.

2.1 Deliverable description

The purpose of this deliverable was to become reasonably proficient at VML. This was to be demonstrated by creating an image of a sunset that uses poly-line's, shape's, oval's, path's, and gradient's. Internet Explorer 5.0 and later is required to display this VML file.

2.2 Readings leading to this deliverable

VML is an XML application for vector graphics that can be embedded in HTML pages where GIF and JPEG images can be appeared. Vector graphics take up less space and thus download much faster than GIF and JPEG over the network [HR01]. VML is natively supported by Internet Explorer 5.0 and later.

The following are a couple of VML shape child elements used in this deliverable:

Element Name	***Purpose***
`Fill`	Specifies how and with what the shape is filled. **Gradient** is an attribute value of Type in element Fill.
`Path`	Commands specifying how to draw the shape's outline.

(Page 1212 of [HR01])

The following is a source code sample that draws a chimney on the browser using `path` and `fill` elements:

```
 <!– draws a chimney on the browser -->
<v:shapetype id="chimney"
    coordsize="10, 3"
    path="M 28,12 L 28, 7 30,7 30,12 X E">
    <v:fill Type="Gradient"  Angle="330"
        color="silver"
        color2="gray"/>
</v:shapetype>
<v:shape type="#chimney" style="width:10; height:3" />
```

"v" is the namespace prefix which binds to the URI `urn:schemas-microsoft-com:vml`. `Shapetype` defines a shape named "chimney", which is made out of a path. The bounding box of this `shapetype` has coordinate size width of 10 and height of 3 in the local coordinate space. "`M 28, 12`" moves the pen to the point x = 28, y = 12. "`L 28, 7 30,7 30,12`" draws a line from the current pen location (x = 28, y = 12) to x = 28, y = 7, then to point x = 30, y = 7, and point x = 30, y = 12. "X" closes the shape by drawing a line from the last point back to the first point. "E" ends the path. The fill element tells the VML renderer to fill this shape with gradient colors silver and gray at a 330 degree angle. The angle 330 specifies the two colors relative position angles along the gradient.

`Oval` and `Polyline` are predefined `Shape` elements employed in this deliverable as well:

Element Name	***Purpose***
`Oval`	The largest oval that can fit in a rectangle box of specified size.
`polyline`	A series of straight lines drawn between successive pairs of points.
`Rect`	A rectangle oriented parallel to the coordinate axes defined by one corner and a height and a width.

(Page 1213 of [HR01])

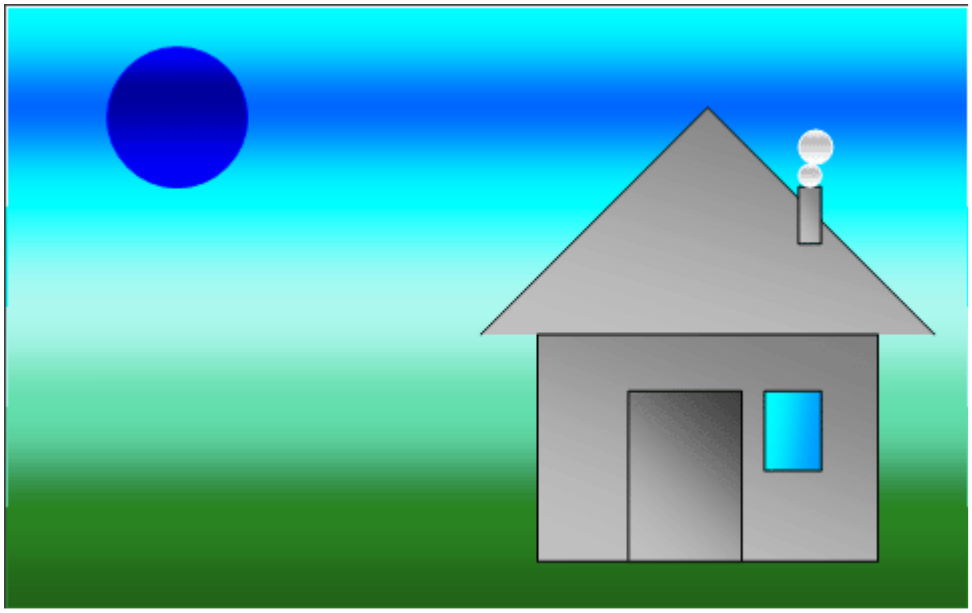
The following is a source code sample that draws a window on the browser using rect element:

```
<!–draws a window -->
<v:rect filled="true" fillcolor="#ff9900"
             style="height:7; width:5; top:25; left:25">
            <v:fill Type="Gradient"  Angle="80"
                   color="#ff9900"
                   color2="yellow"/>
 </v:rect>
```

The above code draws a rectangle that has width of 5 units and height of 7 units. Its left upper corner is at coordinate position x = 25 and y = 25. This rectangle is filled with

gradient colors #ff9900 and yellow at angle 80 degrees. The angle 80 specifies the two colors relative position angles along the gradient.

VML is natively supported by Internet Explorer 5.0 and later. Thus, this deliverable requires an Internet Explorer 5.0 and later to display. Please read Appendix A for the complete source code of this deliverable. The output is shown below:



2.3 How doing this deliverable is going to help in CS 298

VML is this project's target language. The goal of this exercise was to become familiar with some of the basic `Shape` elements, such `as polyline, Oval`, `Path`, and `Fill`, which we will use in the final project. The final product of this project is a transformation from X3D to VML.

3. Deliverable 2: A 3D mock-up apartment in VRML

3.1 Deliverable description

The purpose of this deliverable was to become reasonably proficient at X3D/VRML. This was to be demonstrated by creating a 3D mock-up of an apartment. Nodes used in this deliverable are: `Shape, Appearance, Material, Box, Cone, Cylinder, Sphere, Group, Transform, Rotation, Scaling, Inline`, and others. At this stage of my project, a VRML plugin is still required to display VRML files using Internet Explorer or Netscape web browsers.

3.2 Readings related to this deliverable

Three-dimensional objects in virtual worlds can be built using X3D/VRML(Virtual Reality Modeling Lauguage). A virtual world can be any fragment of the real world you

can think of, for example, a tree, a park, or a city. Predefined primitive shapes: Box, Cone, Cylinder, and Sphere are used to construct these worlds in X3D/VRML. Since there is no free plugins for X3D at this time, I implemented this deliverable in VRML to get myself familiar with X3D. The syntax of X3D and the syntax of VRML are very similar. Examples of the code for a box in X3D and VRML are given below.

A box that has width of 2.0, height of 5.0, and depth of 8.0 in X3D:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE X3D SYSTEM "latest.dtd">
<X3D>
      <Scene>
            <Shape>
                  <Appearance><Material/></Appearance>
                  <Box size="200.0 300.0 400.0" />
            </Shape>
      </Scene>
</X3D>
```

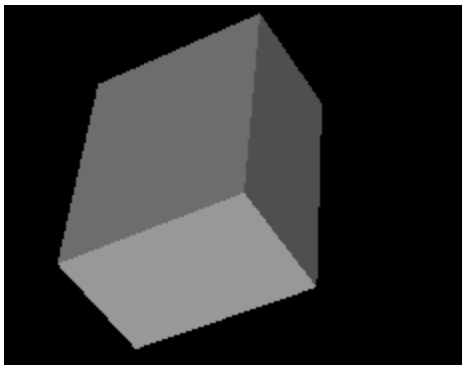
The first line of the above source code states that this file is a standalone version 1.0 XML document. The second line indicates that this XML document is to be validated by a DTD file called "latest.dtd" in the current directory as this file. <X3D> … </X3D> tag are required for all X3D documents. Appearance tag specifies an object's color and surface texture. The default is a glowing white appearance. A shape of a box that has width of 2.0, height of 5.0, and depth of 8.0 is the only object in this world.

A box that has the same dimensions as the above box, but written in VRML.

```
#VRML V2.0 utf8
Shape
{
      appearance Appearance
      { material Material { } }
      geometry Box
      { size 2.0  5.0  8.0 }
}
```

`#VRML V2.0 utf8` is the VRML header, which is required in any VRML file (page 11 of [ANM97]). This box also has a glowing white appearance. `Geometry` specifies object's 3D structure. It could be a `Box, Cone, Cylinder`, or `Sphere` node. Primitive shapes in VRML are always built centered at the origin.

The output of the above box in VRML is shown below:



X3D has multiple formats: tag-based, binary, and VRML, which uses curly braces "{" and "}". The X3D format that I used for this deliverable is tag-based. Even though the syntax between X3D and VRML are different, but their features are the same. I will focus on VRML for the rest of this section.

Each primitive shape has a default value. They are listed below:

Shape Name	***Default Values***
Box	width 2.0 height 2.0 depth 2.0
Cone	radius 1.0 height 2.0
Cylinder	radius 1.0 height 2.0
Sphere	radius 1.0

(page 31 – 34 of [ANM97])

Shapes in VRML can be defined and use later in the code. The following is a source code sample does this.

```
# Left front table leg
Transform
{
    translation -3.5  -3.5  2.5
    children DEF TableLeg Shape
    {
        appearance USE Yellow
        geometry Cylinder
        {
            radius 0.5
            height 7
        }
    }
},
# Right front table leg
Transform
{
    translation 3.5  -3.5  2.5
    children USE TableLeg
},
```

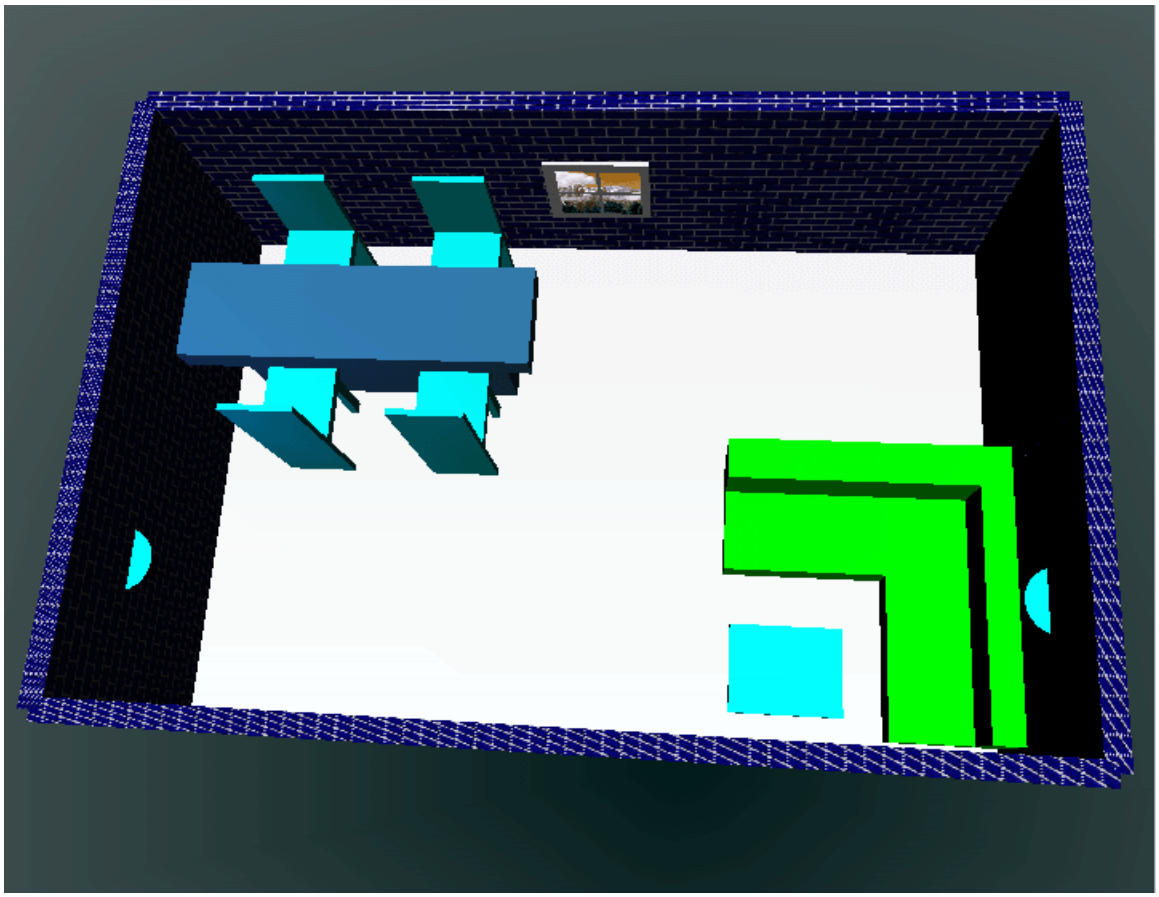
The above `DEF` defines a yellow cylinder called `TableLeg` used in my Deliverable 2, which has radius of 0.5 and height of 7. The original table leg is build at position (-3.5 -3.5 2.5). Then the same table leg is built at position (3.5 -3.5 2.5) by using `USE` command.

The node `Group` groups several Shapes together and can be manipulated together as a group, like `Translation`, `Rotation`, and `Scale`. `Scale` makes the shape smaller or bigger. `Inline` imports shapes into current file. The following is sample using these nodes:

```
Group
{
    children
    [
        # a room with a floor and three walls
        Inline
        {
            url "room.wrl"
        }
        # a bar with a counter and four chairs
        Transform
        {
            translation  -5.75  1.8  -3.0
            scale         1.0   0.75  0.5
            rotation      0.0  1.0   0.0  3.14
            children
            [
                DEF Counter Inline
                {
                   url "counter.wrl"
                }
            ]
        },
       ……
    ]
}
```

The node `Group` has a child node `childen`. It groups all Shapes in the `children` node. `Inline` imports the file “room.wrl” from the current directory into this file. `Scale` scales the height by 75%, the depth by 50%, and the width by 100% of the original shape in “counter.wrl” file. `Rotation` rotates current shape 3.14 radians in y-axes.

At this stage of my project, a VRML plugin needs to be downloaded and installed in order to display this deliverable with Internet Explorer or Netscape. Complete source code of this deliverable can be found at Appendix A. This mock-up apartment has three rooms, a living room, a kitchen, and a bedroom. The following is a view of the living room. Please display this deliverable with a web browser to explore the details.



3.3 How doing this deliverable is going to help in CS 298

The goal of this project is to transform X3D to VML. Since there is no free plugins for X3D at this time, I implemented this deliverable in VRML to get myself familiar with X3D. The syntax of X3D and the syntax of VRML are very similar as illustrated by the code for a box in X3D and VRML are shown in section 3.2. VRML has many features besides the nodes introduced here. My project will not translate all the nodes introduced above. My project will translate the core nodes, such as `Box`, `Cylinder`, `Cone`, `Sphere`, `Translate`, `Rotation`, and `Scale`. I will add more features to my project depending on the difficulty level of other features and time constraints.

4. Deliverable 3: Histograms translated to VML

4.1 Deliverable description

This deliverable contains an XML DTD for histogram and an XSL transformation from this DTD to VML.

4.2 Readings leading to this deliverable

4.2.1 XML

XML (Extensible Markup Language) is a subset of SGML (Standard Generalized Markup Language). XML eliminates much of the complexity of SGML, but has the same goal as SGML. XML can mark up any type of data, for example Scalable Vector Graphics (SVG), MathML, and Chemical Markup Language (CML). XML describes a syntax, which you use to create your own languages [HD00].

Rules for elements (page 31 of [HD00]):

- Every start-tag must have a matching end-tag
- Tags cannot overlap
- XML documents can have only one root element
- Element names must obey XML naming conventions
- XML is case-sensitive
- XML will keep white space in your text

The following is a code fragment from file “histogram.xml”. It meets all the rules for a well-formed XML file.

```
<!-- top element, histograms may contains 0 or more histogram tags
-->
<histograms>
<!-- a histogram may contain 1 or more bins,
     binwidth specifics bin width
     maxNumber specifics the maximum bin height
-->
<histogram binwidth="15" maxNumber="100">
    <!-- number specifics height of each bin -->
    <bin number="100" />
    <bin number="0" />
    <bin number="70" />
    <bin number="88" />
    <bin number="90" />
</histogram>
</histograms>
```

The above code contains a root element "histograms". A "histograms" may contain zero or more "histogram", and a "histogram" must contain 1 or more bins.

4.2.2 **DTD**

The purpose of a Document Type Definition is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.

The following is the DTD verifies the “histogram.xml” above:

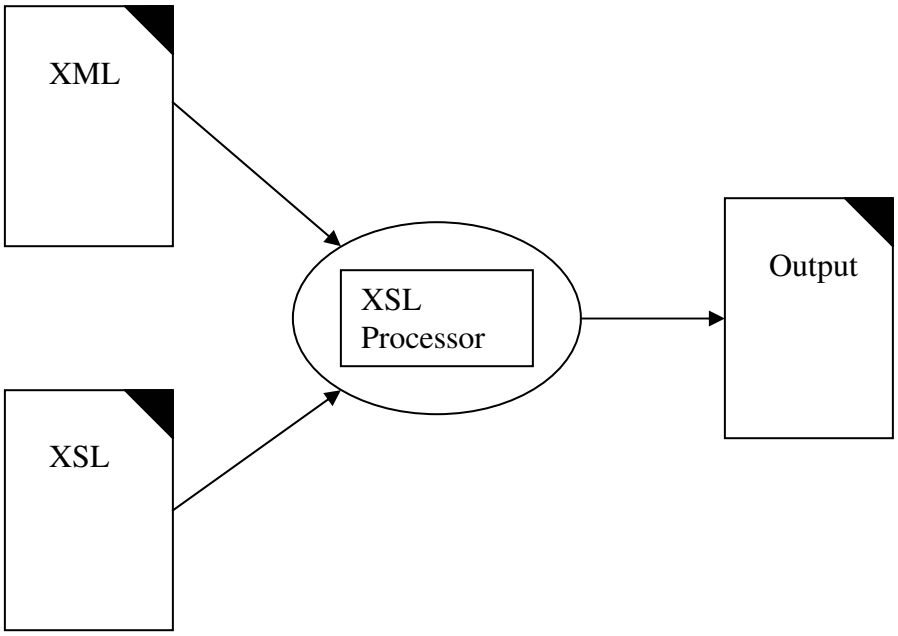
```
<!-- histograms contains 0 or more histogram -->
<!ELEMENT histograms (histogram) >
<!-- histogram contains 1 or more bins -->
<!ELEMENT histogram (bin+) >
<!-- histogram's binwidth and maxNumber attribute can not be null
-->
<!ATTLIST histogram
    binwidth    CDATA   #REQUIRED
```

```
    maxNumber   CDATA   #REQUIRED
>
<!-- bin tag contains no data, only attribute number -->
<!ELEMENT bin EMPTY>
<!-- bin tag contains only attribute number -->
<!ATTLIST bin
    number      CDATA   #REQUIRED
>
```

The "histogramDTD.dtd" validates "histogram.xml" to ensure all elements and attribute are valid. For example, histogram's binwidth and maxNumber attribute can not be null, bin tag contains only attribute number.

4.2.3 XSLT

XML documents usually contain data only. They do not include any formatting information. The information in an XML document may not be in the form in which it is desired to present it. XSL provides information on how to present or process the XML document. You may view their relations as below:



My project mainly uses XSLT, which describes how to transform one XML document into another. XSLT stylesheets are built on template structures. A template specifies what to look for in the source tree, and what to put into the result tree. The following is a code fragment from file “histogram.xsl”:

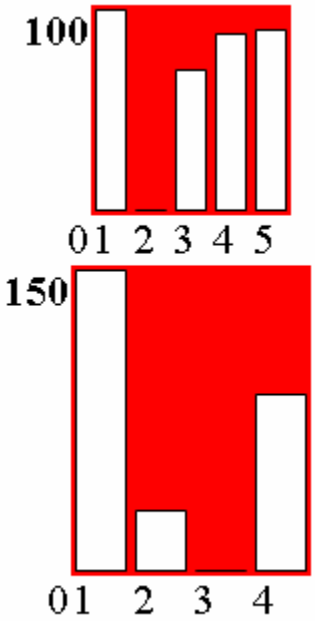
```
<!-- output a table for every histogram tag encountered -->
<xsl:template match="histogram">
    <center>
    <!-- sets table cellspacing and border to 0 -->
```

```
<table cellspacing="0" border="0">
        <tr>
                    <!-- output max column number -->
              <th valign="top" align="right" bgcolor="white">
                    <xsl:value-of select="@maxNumber"/>
              </th>
                    <!-- output a bin for each bin tag -->
                    <xsl:apply-templates/>
         </tr>
    …
    …
    …
    …

    </table>
    </center>
</xsl:template>
```

The above template outputs a table at the center for each histogram tag it encounters. The table header value "th" for the output document is the `maxNumber` attribute value from the current tag of the input document. Apply further templates for each bin tag it encounters.

The transformed document of this deliverable is in VML. Internet Explorer 5.0 and later is required to view the output. Please read Appendix A for complete source code of this deliverable. The output is shown below:



4.3 Tricky coding

The most tricky part of this deliverable is to output VML rectangle tags from the stylesheet. A VML rectangle tag sample is shown below, which is what I wanted in the output file:

```
<v:rect style=”height: 100; width: 20; top: 0; left: 0” filled=”true”
filledcolor=”white”/>
```

`v` is the namespace to the URI `urn:schemas-microsoft.com:vml`. A white rectangle with width of 20 units and height of 100 units is drawn on the left upper corner position at x = 0 and y = 0. XSLT processor does not allow creating the above rect tag directly. First of all, a tag name cannot contain a colon “:”. Secondly, XSLT processor does not allow multiple attribute values for attribute `style`. What I did is to output this `rect` tag separately. A `rect` tag is broken down into eight XSLT tags. Please see the source code below on how I did it:

```
<!-- output rectangle tag and its attributes for each bin -->
<!-- output rectangle's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of rectangle attribute, 'v:rect style="' -->
<xsl:text disable-output-escaping="yes">v:rect style=&quot;</xsl:text>
<!-- output part of rectangle attribute name height -->
<xsl:text disable-output-escaping="yes">height:</xsl:text>
<!-- output rectangle's height -->
<xsl:value-of select="@number"/>
<!-- output rectangle's width attribute name -->
<xsl:text disable-output-escaping="yes">; width:</xsl:text>
<!-- output rectangle's width -->
<xsl:value-of select="../@binwidth"/>
<!-- set rectangle's top and left position to 0, and end rectangle tag -
->
<xsl:text disable-output-escaping="yes">; top:0; left:0&quot;</xsl:text>
<!-- set bins color to white and close rect tag -->
<xsl:text disable-output-escaping="yes">filled=&quot;true&quot;
fillcolor=&quot;white&quot; /&gt;</xsl:text>
```

4.4 How this deliverable is going to help in CS 298

This deliverable transforms histograms to VML using XSLT. My project is also a transformation to VML using XSLT plus Javascript. The transformation for my project is much more difficult. This is just a practice of a simple transformation.

5. Deliverable 4: Box in VML

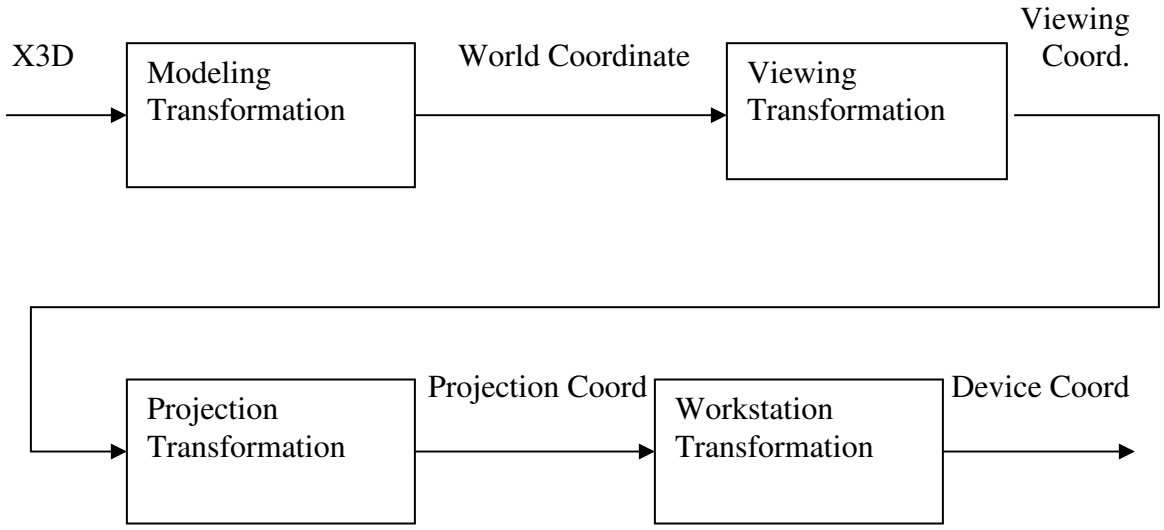
5.1 Deliverable description

This deliverable outputs a box’s wireframe in VML using XSLT and Javascript. The input file is in X3D.

5.2 Readings leading to this deliverable

5.2.1 Graphical Concepts

For this deliverable, we need to know some of the graphical concepts first. VRML is in three demensional. VML is in two dimensional. The first step is to get the world coordinates of the object in 3D. The second step to obtain the viewing coordinates. The third step is to project the resulting viewing coordinates into projection coordinates.



(page 433 of [HB97])

X3D specifies a box's dimension by its width, height, and depth. From a box's width, height, and depth, we can calculate its eight points world coordinates. `Box` is one of X3D's primitive shapes. So it is centered at the origin by default. For now, let our viewing coordinate be the same as world coordinate. Then, perform a projection of the resulting viewing coordinates. Perspective projection is chosen because perspective projection produces realistic views.

5.2.2 Javascript

Extension functions in another language such as Java or JavaScript can be embeded in an XSLT document to extend its capabilities. The common reasons for doing this are:

- To improve performance (for example when doing complex string manipulations)
- To exploit system capabilities and services
- To reuse code that already exists in another language
- For convenience, as complex algorithms and computations can be very verbose when written in XSLT

(page 132 of [KM01])

The reason that I used JavaScript in my stylesheet is to write my own functions for computations. The following Javascript is taken from my Deliverable 4 stylesheet document.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt"
    xmlns:helper="http://deliverable4">

      <msxsl:script language="Javascript" implements-prefix="helper">
            <![CDATA[
```

```
            var dz = 100;

            /**

                  Returns x coordinate given the box's width and
                  index of this point.
            */
            function xVC(size, index)
            {
                  // left to the origin
                  if(index == 1 || index == 4 || index == 5 ||
                  index == 6)
                  {
                        return 0 - size/2;
                  }
                  // right to the origin
                  else if(index == 2 || index == 3 || index == 7
                  || index == 8)
                  {
                        return 0 + size/2;
                  }
            }
            . . .

      ]]>

</msxsl:script>
```

To embed Javascript in an XSLT document, a namespace needs to be declared first. "`Helper`" is declared as a namespace for the code above. "`JavaScript`" indicates the following script will be written in Javascript. The name for `implements-prefix` must match the namespace, in this case is "helper". Everything inside `<![CDATA[ ]]>` is Javascript. The above Javascript first declares a variable called dz with initial value of 100. The function `xVC( )` returns x coordinate position given the box's width and index of this point.

The following are the main steps my stylesheet performed to transform a box from X3D to VML:

A. Parse the box's width, heigth, and depth from X3D's Box node:

```
<!-- gets box's sizes from box tag -->
<xsl:variable name="boxDim" select="@size"/>
<xsl:variable name="x" select="substring-before($boxDim, ' ')"/>
<xsl:variable name="rest" select="substring-after($boxDim, ' ')"/>
<xsl:variable name="y" select="substring-before($rest, ' ')"/>
<xsl:variable name="z" select="substring-after($rest, ' ')"/>
```

The syntax of X3D's Box node is: <Box size="200.0 200.0 200.0" />. Its width, height, and depth are all 200.0 in this case. To get its width, we get the first token before the first space of the attribute "size". Its height and depth are the second and third token respectively.

B. Obtain the view coordinate position of a point in x, y, and z axes:

```
<xsl:variable name="xVC1" select="helper:xVC($x, 1)" />
```

```
<xsl:variable name="yVC1" select="helper:yVC($y, 1)" />
<xsl:variable name="zVC1" select="helper:zVC($z, 1)" />
```

The above source code calculates the view coordinate of Point 1 in the x, y, and z axes and stores these values in variable xVC1, yVC1, and zVC1 respectively. Functions xVC( ), yVC( ), and zVC( ) are defined in Javascript in namespace "helper". The second argument of these functions are the index of this point, which range from 1 to 8.

C. Get the projected x and y coordinate:

```
<xsl:variable name="xP1" select="helper:projectionPos($xVC1,
$zVC1)" />
<xsl:variable name="yP1" select="helper:projectionPos($yVC1,
$zVC1)" />
```

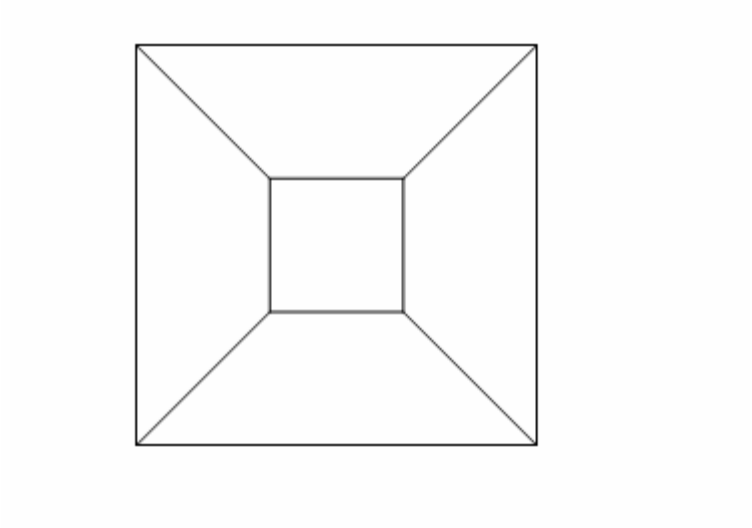
Function projectionPos( ) returns the projected value of either x or y given in the first argument. The second argument is its z-axis value. The above source code calculates the projected x and y coordinates and store their values in xP1 and yP1.

D. Form a line between two points in VML:

```
<!--link point 1 and 2 -->
<!-- output line tag and its attributes for each line -->
<!-- output line's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of line attribute, 'v:line' -->
<xsl:text disable-output-escaping="yes">v:line </xsl:text>
<!-- output part of line attribute from -->
<xsl:text disable-output-escaping="yes">from="</xsl:text>
<!-- output rectangle's from attribute for x and y -->
<xsl:value-of select="$xP1"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP1"/>
<!-- output rectangle's to attribute for x and y -->
<xsl:text disable-output-escaping="yes">" to="</xsl:text>
<!-- output rectangle's width -->
<xsl:value-of select="$xP2"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP2"/>
<!-- close line tag -->
<xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>
```

The above XSLT code splits VML code that draws a line between Point 1 and Point 2.

The output of this deliverable is in VML. So, Internet Explorer 5.0 and later is required to display. A screen dump of a box that has width, height and depth of 200.0 is shown below:



5.3 Challenges of this deliverable

5.4 There are two challenges for this deliverable. The first challenge is to learn about the graphical concepts. The second challenge is to get Javascript to work inside the stylesheet.

5.5 How doing this deliverable is going to help in CS 298

This deliverable takes me one big step closer to my final project. My XSLT stylesheet outputs the wireframe of a box in VML. The next step is to handle shading. Then add more features, such as `Translation, Scale`, and `Rotation`. After finish working with `Box, Cone, Cylinder,` and `Sphere` will be translated also from X3D to VML.

6 Challenges for my project

There are five challenges for my project:

I. VML only supports very simple basic tags, such as `Shape, line, oval, polyline`, etc. My project needs to leave hooks so that different orientations of the object can still be computed.

II. This translator needs to be written efficiently so it will not take for a long time to be downloaded over the busy network.

III. An efficient painting method needs to be applied if multiple objects exist.

IV. An efficient data structure is required to keep track of the relation between points, lines, and polygons. Cone, Cylinder, and Sphere are made out of many small polygons. The all require such a data structure.

V. VML supports gradient. So I will use gradient for shading. For complicated shapes, such as Sphere, Cone, and Cylinder, shading is very challenging. For

example, to represent a circle, I will use many small triangles. I need to come up a method to keep track of each triangle relative to the viewpoint, so I will apply gradient shading to each triangle.

7. Work have completed so far and future work overview

7.1 Work have done so far

I have studied VML, VRML, XSLT, Javascript and some graphical concepts during the Spring 2002 semester. It helps me to visualize the challenges that I will face and proven that my project is feasible. The five challenges for my project are listed in section 6. Deliverable 1 helps me to become reasonably proficient at VML, which is my project target language. Deliverable 2 explores some VRML basic features, which is my project input language's super language. Deliverable 3 gives me a chance to learn XSLT and learn the tricks that I need to play to output some VML tags. Deliverable 4 outputs a wireframe of a box in VML using XSL and Javascript. This deliverable takes me to the door of my final project product.

7.2 Future work overview

My XSLT stylesheet in Deliverable 4 outputs the wireframe of a box in VML. The next step is to shade the box. Then add more features, such as `Translation`, `Scale`, and `Rotation`. After finish working with `Box` node, `Cone` node, `Cylinder` node, and `Sphere` node will be also translated from X3D to VML. More X3D nodes might be translated depending on the difficulty level and time constraints.

References:

[ANM97] VRML 2.0 Sourcebook. 2nd edition. A. Ames, D. Nadeau, J. Moreland. Wiley. 1997.

[ECMA99] Standard ECMA-262 ECMAScript Language Specification 3rd ed. http://www.ecma.ch/ecma1/stand/ecma-262.htm. ECMA. 1999.

[FD98] JavaScript: The Definitive Guide. 3rd edition. David Flanagan. O'Reilly & Associates. 1998.

[GR00] 3D Interactive VML. http://www.gersolutions.com/vml/. Gareth Richards. 2000.

[HB97] Computer Graphics C Version. 2nd edition. Donald Hearn, M. Pauline Baker. Prentic Hall. 1997.

[HD00] Beginning XML. David Hunter, with Curt Cagle, Dave Gibbons, Nikola Ozu, Jon Pinnock, Paul Spencer. Wrox Press Ltd. 2000

[HR01] Xml Bible. 2nd edition. Elliotte Rusty Harold. Hungry Minds, Inc. 2001.

[KM01] XSLT Programmer's Reference. 2nd edition. Michael Kay. Wrox Press Ltd. 2001.

[MK01] LiveGraphics3D Documentation. http://wwwvis.informatik.uni-stuttgart.de/~kraus/LiveGraphics3D/documentation.html. Martin Kraus. 2001.

[RE01] Learning XML. Erik T. Ray. 1st edition. O'Reilly & Associates. 2001.

[W00] 3D Computer Graphics. 3rd ed. Allan Watt. Pearson Education Limited. (Addison Wesley). 2000.

[W3C01] Scalable Vector Graphics (SVG) Specification 1.0. http://www.w3.org/TR/SVG/. W3C.

[W3C97] Extensible Markup Language (XML). http://www.w3.org/XML. W3C.

[W3C98b] VML - the Vector Markup Language. http://www.w3.org/TR/NOTE-VML. W3C.

[W3C99] XSL Transformations (XSLT) Version 1.0. http://www.w3.org/TR/xslt. W3C.

[W3DC01] Extensible 3D (X3D) Graphics Working Group. http://www.web3d.org/x3d.html. Web 3D Consortium. 2001

Appendix A

I. Deliverable 1

"hw1d.html"

```
<html xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns="http://www.w3.org/TR/REC-html40">

<head>

<!--[if !mso]>
<style>
v\:* {behavior:url(#default#VML);}
o\:* {behavior:url(#default#VML);}
.shape {behavior:url(#default#VML);}
</style>
<![endif]-->
```

```
</head>

<body>

<!-- for background -->
<v:rect
  fill="false"
  stroke="false"
  strokecolor="white"
  strokeweight="0.75pt"
  border="0"
  style="position:absolute;
  top:0px; left:0;
  width:480px;
  height:100px">
  <v:fill Type="Gradient"
    Angle="360"
    color="#ff6600"
    color2="yellow"
    Focus="50%" />
</v:rect>
<v:rect
  fill="true"
  stroke="true"
  strokecolor="yellow"
  strokeweight="0.75pt"
  border="0"
  border="0"
  style="position:absolute;
  top:100px; left:0;
  width:480px;
  height:50px">
  <v:fill Type="Gradient"
    Angle="360"
    color="yellow"
    color2="#EFF8Ad"/>
</v:rect>
<v:rect
  fill="true"
  stroke="true"
  strokecolor="#EFF8Ad"
  strokeweight="0.75pt"
  style="position:absolute;
  top:150px; left:0;
  width:480px;
  height:50px">
  <v:fill Type="Gradient"
    Angle="360"
    color="#EFF8Ad"
    color2="#ADDE63"
     />
</v:rect>
<v:rect
  fill="true"
  stroke="true"
  strokecolor="#ADDE63"
  strokeweight="0.75pt"
  style="position:absolute;
  top:200px; left:0;
  width:480px;
  height:50px">
  <v:fill Type="Gradient"  Angle="360"
    color="#ADDE63"
```

```
    color2="#218429"
     />
</v:rect>
<v:rect
  fill="true"
  stroke="true"
  strokecolor="#218429"
  strokeweight="0.75pt"
  style="position:absolute;
  top:250px; left:0;
  width:480px;
  height:50px">
  <v:fill Type="Gradient"  Angle="360"
    color="#218429"
    color2="#186321"
     />
</v:rect>

<!-- group components for the house together -->
<v:group id="house" style="position:absolute; top:50px;
left:350;Swidth:6cm; height:6cm"
    coororigin="0,0" coordsize="40, 40">
    <!-- roof -->
    <v:polyline filled="true" fillcolor="gray"
        points="0, 20, 20, 0, 40, 20">
        <v:fill Type="Gradient"  Angle="330"
            color="silver"
            color2="gray"/>
    </v:polyline>

    <!-- chimmy -->
    <v:shapetype id="chimmy"
        coordsize="10, 3"
        fillcolor="black"
        path="M 28,12 L 28, 7 30,7 30,12 X E">
        <v:fill Type="Gradient"  Angle="330"
            color="silver"
            color2="gray"/>
    </v:shapetype>
    <v:shape type="#chimmy" style="width:10; height:3" />

    <!-- small smoke -->
    <v:oval id="smoke1" filled="true" fillcolor="#ff3300"
        stroke="true"
        strokecolor="white"
        strokeweight="0.75pt"
        style="height:2; width:2; top:5; left:28">
    <v:fill Type="Gradient"
        color="white"
        color2="silver"
        focus="30%" />
    </v:oval>

    <!-- bigger smoke -->
    <v:oval id="smoke2" filled="true" fillcolor="#ff3300"
        stroke="true"
        strokecolor="white"
        strokeweight="0.75pt"
        style="height:3; width:3; top:2; left:28">
    <v:fill Type="Gradient"
        color="white"
        color2="silver"
        focus="30%" />
    </v:oval>
```

```
    <!-- wall -->
    <v:rect filled="true" fillcolor="gray"
        style="height:20; width:30; top:20; left:5">
        <v:fill Type="Gradient"  Angle="330"
            color="silver"
            color2="gray"/>
    </v:rect>

    <!-- window -->
    <v:rect filled="true" fillcolor="#ff9900"
        style="height:7; width:5; top:25; left:25">
        <v:fill Type="Gradient"  Angle="80"
            color="#ff9900"
            color2="yellow"/>
    </v:rect>

    <!-- door -->
    <v:rect filled="true" fillcolor="black"
        style="height:15; width:10; top:25; left:13">
        <v:fill Type="Gradient"  Angle="330"
            color="silver"
            color2="#404040"/>
    </v:rect>
</v:group>

<!-- sun -->
<v:oval filled="true" fillcolor="#ff3300"
    stroke="true"
    strokecolor="red"
    strokeweight="0.75pt"
    style="position:absolute; top:20px; left:50;
width:70px;height:70px">
<v:fill Type="Gradient"
    color="red"
    color2="#990000"
    focus="30%" />
</v:oval>

</body>

</html>
```

II. Deliverable 2

“allWorld.wrl”

```
# the following code has all world files that build
# the apartment.  They are in alphabetical order.

#VRML V2.0 utf8
# file name: bedRoom.wrl
# A bedroom with a desk that has a lamp on it, and a bed
Group
{
    children
    [
        # a room with a floor and three walls
        Inline
        {
            url "room.wrl"
        }
        # a bed with two red pillows
```

```
Transform
{
    translation  -7.0  0.5  -4.5
    children
    [
        # builds a bed with a box
        Shape
        {
            appearance DEF BedMaterial Appearance
            {
                material Material
                {
                    diffuseColor 1.0  0.9  0.83
                }
            }
            geometry DEF Bed Box
            {
                size 7.0  1.0  5.0
            }
        }
        # rear pillow
        Transform
        {
            translation  -2.0 0.65  -1.0
            scale         0.2  0.2   0.2
            rotation      0.0  1.0   0.0 -1.57
            children
            [
                Shape
                {
                    appearance DEF PillowAppearance Appearance
                    {
                        material Material
                        {
                            diffuseColor 0.8  0.0  0.0
                        }
                    }
                    geometry USE Bed
                }
            ]
        }
        # front pillow
        Transform
        {
            translation -2.0 0.65  1.0
            scale         0.2  0.2   0.2
            rotation      0.0  1.0   0.0 -1.57
            children
            [
                Shape
                {
                    appearance USE PillowAppearance
                    geometry USE Bed
                }
            ]
        }
    ]
},
# a chair next to the desk
Transform
{
    translation   6.5  2.3  3.0
    scale         0.2  0.33  0.3
    rotation      0.0  1.0  0.0 1.57
```

```
            children
            [
                DEF Chair Inline
                {
                   url "chair.wrl"
                }
            ]
        },
        # a study desk with a lamp
        Transform
        {
            translation   8.75 3.0  3.0
            scale         0.5  0.4  0.5
            rotation      0.0  1.0  0.0 1.57
            children
            [
                # use a table as a study desk
                Inline
                {
                   url "table.wrl"
                }
                Transform
                {
                    translation   2.0   0.55  1.0
                    scale         10.0  10.0  10.0
                    rotation      0.0    1.0  0.0 3.14
                    children
                    [
                        # a lamp placed on the table
                        Inline
                        {
                            url "lamp.wrl"
                        }
                    ]
                },
            ]
        },

    ]
}


#VRML V2.0 utf8
# file name: bedRoomVP.wrl - bedroom with view point
# kitchen with viewpoint set
Group
{
    children
    [
        # sets viewpoint as forward view, initial position and
orientation
        Viewpoint
        {
            description "Forward view"
            position      0.0  6.6  0.0
            orientation   0.0  1.0  0.0  3.14
        }
        # sets avator size, walk movement type, walking speed,
        # and with headlight on
        NavigationInfo
        {
            type "WALK"
            speed 1.0
            headlight TRUE
```

```
            avatarSize [ 0.5  1.6  0.5 ]
        }
        # a bedroom with a bed, a desk, and a chair
        Inline { url "bedRoom.wrl" }
        # create a door that acts as an anchor
        # to the kitchen
        Anchor
        {
            url "livingRoomVP.wrl"
            description "To the kitchen"
            children
            [
                # a white door
                Transform
                {
                    translation  -10.25  0.0  1.0
                    rotation     0.0  1.0  0.0  1.57
                    children Inline { url "door.wrl" }
                }
            ]
        },
    ]
}


#VRML V2.0 utf8
# file name: chair.wrl
# A yellow chair
Group
{
    children
    [
        # seat which has two legs and a seat
        Inline
        {
            url "chairSeat.wrl"
        }
        # chair back using a box
        # acts a chair's back and two rear legs
        Transform
        {
            translation  0.0  0.5  -3.25
            children
            [
                Shape
                {
                    appearance Appearance
                    {
                         material Material
                        {
                            diffuseColor 1.0  1.0  0.0
                        }
                    }
                    geometry Box
                    {
                        size 9 15 0.5
                    }
                }
            ]
        },

    ]
}
```

```
#VRML V2.0 utf8
# file name: chairSeat.wrl
# chair seat which has two legs and a seat
Group
{
    children
    [
        # chair seat
        Shape
        {
            appearance DEF Yellow Appearance
            {
                material Material
                {
                    diffuseColor 1.0  1.0  0.0
                }
            }
            geometry Box { size 9  1.0  7 }
        },
        # Left front chair leg
        Transform
        {
            translation -3.5  -3.5  2.5
            children DEF TableLeg Shape
            {
                appearance USE Yellow
                geometry Box
                {
                    size 1.0  7  1.0
                }
            }
        },
        # Right front chair leg
        Transform
        {
            translation 3.5  -3.5  2.5
            children USE TableLeg
        },
    ]
}

#VRML V2.0 utf8
# file name: counter.wrl
# a counter simply built with two boxes
Group
{
    children
    [
        # counter stand built with a box
        Shape
        {
            appearance DEF Wood Appearance
            {
                material Material
                {
                    diffuseColor 0.7  0.5  0.2
                }
            }
            geometry Box { size 7  5  3 }
        },
        # counter top built with a box
        Transform
        {
```

```
            translation -0.15  2.5  0.0
            children Shape
            {
                appearance USE Wood
                geometry Box { size 9  1.0  5.0 }
            }
        },
    ]
}


#VRML V2.0 utf8
# file name: door.wrl
# build a plain white door with a gray handle
Group
{
    children
    [
        # white door built with a box
        Transform
        {
            translation 0.0  3.0  0.0
            children
            [
                Shape
                {
                        appearance Appearance
                        {
                            material Material
                            {
                                diffuseColor 1.0  1.0  1.0
                            }
                        }
                        geometry Box
                        {
                            size 3.0  6.0  0.5
                        }
                }
          ]
        },
        # door handle built with a gray sphere
        Transform
        {
            translation -1.0  3.0  0.4
            children
            [
                Shape
                {
                        appearance Appearance
                        {
                            material Material
                            {
                                diffuseColor 0.5  0.5  0.5
                            }
                        }
                        geometry Sphere
                        {
                            radius 0.2
                        }
                }
          ]
        },
    ]
}
```

```
#VRML V2.0 utf8
# file name: kitchen.wrl
# A kitchen with wash counter, stove which has two cooking pots on it,
# a refrigerator, a dining table, and two chairs
Group
{
    children
    [
        # a room with a floor and three walls
        Inline
        {
            url "room.wrl"
        }
        # a stove counter
        Transform
        {
            translation  -7.125  1.2  -6.0
            scale         0.75   0.5  0.5
            children
            [
                DEF Counter Inline
                {
                   url "counter.wrl"
                }
            ]
        },
        # large pot on the left
        Transform
        {
            translation  -7.125  3.2  -5.75
            scale         0.5   0.5  0.5
            children
            [
                DEF Pot Inline
                {
                   url "pot.wrl"
                }
            ]
        },
        # small pot on the right
        Transform
        {
            translation  -5.125  3.2  -5.75
            scale         0.3   0.3  0.3
            children USE Pot
        },
        # a wash counter
        Transform
        {
            translation  -9.25  1.2  -3.0
            scale         0.75   0.5  0.5
            rotation     0.0  1.0  0.0  1.57
            children USE Counter
        },
        # a frigerator
        Transform
        {
            translation   9.5  2.5  -5.75
            scale         1.0  1.0  1.0
            rotation      0.0  1.0  0.0  -1.57
            children
            [
```

```
                Inline
                {
                   url "refrigerator.wrl"
                }
            ]
        },
        # a dining table
        Transform
        {
            translation   5.5  3.5  2.75
            scale         0.5  0.5  0.5
            children
            [
                Inline
                {
                   url "table.wrl"
                }
            ]
        },
        # a dining chair on the right of table
        Transform
        {
            translation   7.5  2.5  2.75
            rotation      0.0  1.0  0.0  -1.57
            scale         0.2  0.35  0.3
            children
            [
                DEF Chair Inline
                {
                   url "chair.wrl"
                }
            ]
        },
        # a dining chair on the left the table
        Transform
        {
            translation   3.5  2.5  2.75
            rotation      0.0  1.0  0.0  1.57
            scale         0.2  0.35  0.3
            children USE Chair
        },
    ]
}


#VRML V2.0 utf8
# file name: kitchenVP.wrl - kitchen with viewpoint set
# kitchen with viewpoint set
Group
{
    children
    [
        # sets viewpoint as forward view, initial position and
orientation
        Viewpoint
        {
            description "Forward view"
            position 0.0  6.6  0.0
            orientation   0.0  1.0  0.0  3.14
        }
        # sets avator size, walk movement type, walking speed,
        # and with headlight on
        NavigationInfo
        {
```

```
            type "WALK"
            speed 1.0
            headlight TRUE
            avatarSize [ 0.5  1.6  0.5 ]
        }
        # a kitchen with a wash counter, stove which has
        # two cooking pots on it, a refrigerator, a table
        # and two chairs next to it.
        Inline { url "kitchen.wrl" }
        # create a door that acts as an anchor
        # to the bedroom
        Anchor
        {
            url "bedRoomVP.wrl"
            description "To the kitchen"
            children
            [
                # a white door with a gray handle
                Transform
                {
                    translation  10.25  0.0  -3.0
                    rotation     0.0  1.0  0.0  -1.57
                    children Inline { url "door.wrl" }
                }
            ]
        },
    ]
}


#VRML V2.0 utf8
# file name: lamp.wrl
# The VRML 2.0 Sourcebook
# Copyright [1997] By
# Andrea L. Ames, David R. Nadeau, and John L. Moreland
Group {
  children [
  # Lamp
    DEF MoveLamp PlaneSensor { },
    DEF Lamp Transform {
      children [
      # Lamp base
        Shape {
          appearance DEF White Appearance {
            material Material { }
          }
          geometry Cylinder {
            radius 0.1
            height 0.01
          }
        },
      # Base - First arm joint
        Group {
          children [
            DEF MoveFirstArm SphereSensor {
              offset 1.0 0.0 0.0 -0.7
            },
            DEF FirstArm Transform {
              translation 0.0 0.15 0.0
              rotation    1.0 0.0 0.0  -0.7
              center      0.0 -0.15 0.0
              children [
              # Lower arm
                DEF LampArm Shape {
```

```
                appearance USE White
                geometry Cylinder {
                  radius 0.01
                  height 0.3
                }
              },
            # First arm - second arm joint
              Group {
                children [
                  DEF MoveSecondArm SphereSensor {
                    offset 1.0 0.0 0.0 1.9
                  },
                  DEF SecondArm Transform {
                    translation 0.0 0.3 0.0
                    rotation  1.0 0.0 0.0  1.9
                    center    0.0 -0.15 0.0
                    children [
                    # Second arm
                      USE LampArm,
                    # Second arm - shade joint
                      Group {
                        children [
                          DEF MoveLampShade SphereSensor {
                            offset 1.0 0.0 0.0 -1.25
                          },
                          DEF LampShade Transform {
                            translation 0.0 0.075 0.0
                            rotation  1.0 0.0 0.0  -1.25
                            center    0.0 0.075 0.0
                            children [
                            # Shade
                              Shape {
                                appearance USE White
                                geometry Cone {
                                  height 0.15
                                  bottomRadius 0.12
                                  bottom FALSE
                                }
                              },
                            # Light bulb
                              Transform {
                                translation 0.0 -0.05 0.0
                                children Shape {
                                  appearance USE White
                                  geometry Sphere {
                                    radius 0.05
                                  }
                                }
                              }
                            ]
                          }
                        ]
                      }
                    ]
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
]
```

```
}
ROUTE MoveLamp.translation_changed   TO Lamp.set_translation
ROUTE MoveFirstArm.rotation_changed  TO FirstArm.set_rotation
ROUTE MoveSecondArm.rotation_changed TO SecondArm.set_rotation
ROUTE MoveLampShade.rotation_changed TO LampShade.set_rotation


#VRML V2.0 utf8
# file name: light.wrl
# a light built with a cone upside down, and a pointlight
Group
{
    children
    [
        # light built with a cone upside down
        Transform
        {
            rotation  1.0  0.0  0.0  3.14
            children
            [
                Shape
                {
                    appearance Appearance
                    {
                        material Material
                        {
                            diffuseColor 1.0  1.0  0.0
                        }
                    }
                    geometry Cone
                    {
                        bottomRadius 0.75
                        height       2.0
                    }
                }
            ]
        },
        # a pointlight on top of the cone
        Transform
        {
            translation 0.0  1.0  0.0
            children
            [
                PointLight
                {
                    location 0.0  2.0  0.0
                    color    1.0  0.4  0.2
                    intensity 0.8
                    attenuation 0.0  0.6  0.0
                    radius 10.0
                }
            ]
        }
    ]
}


#VRML V2.0 utf8
# file name: livingRoom.wrl
# A living room with a bar that has four chairs,
# sofa, and a tea table
Group
{
    children
```

```
[
    # a room with a floor and three walls
    Inline
    {
        url "room.wrl"
    }
    # a bar with a counter and four chairs
    Transform
    {
        translation  -5.75  1.8  -3.0
        scale         1.0   0.75  0.5
        children
        [
            DEF Counter Inline
            {
               url "counter.wrl"
            }
        ]
    },
    # a bar chair on the rear left of the bar
    Transform
    {
        translation   -7.5  2.5  -5.0
        scale         0.2  0.35  0.3
        children
        [
            DEF Chair Inline
            {
               url "chair.wrl"
            }
        ]
    },
    # a bar chair on the rear right of the bar
    Transform
    {
        translation   -3.5  2.5  -5.0
        scale         0.2  0.35  0.3
        children USE Chair
    },
    # a bar chair on the front left of the bar
    Transform
    {
        translation   -7.5  2.5  -1.0
        scale         0.2  0.35  0.3
        rotation      0.0  1.0   0.0  3.14
        children
        [
            DEF Chair Inline
            {
               url "chair.wrl"
            }
        ]
    },
    # a bar chair on the front right of the bar
    Transform
    {
        translation   -3.5  2.5  -1.0
        scale         0.2  0.35  0.3
        rotation      0.0  1.0   0.0  3.14
        children USE Chair
    },

    # a tea table
    Transform
```

```
        {
            translation   5.0  1.5  5.0
            scale         0.3  0.2  0.3
            children
            [
                Inline
                {
                   url "table.wrl"
                }
            ]
        },
        # a sofa in the middle of the living roon
        Transform
        {
            translation   7.0  0.5  1.0
            children
            [
                DEF Sofa Inline
                {
                   url "sofa.wrl"
                }
            ]
        },
        # sofa against the wall
        Transform
        {
            translation   9.0  0.5  3.0
            rotation      0.0  1.0  0.0  -1.57
            children USE Sofa
        },
    ]
}


#VRML V2.0 utf8
# file name: livingRoomVP.wrl - living room with viewpoint
# living room with viewpoint set
# adds viewpoint and a door to "livingRoom.wrl"
Group
{
    children
    [
        # sets viewpoint as forward view, initial position and
orientation
        Viewpoint
        {
            description "Forward view"
            position 0.0  7.6  0.0
            orientation   0.0  1.0  0.0  3.14
        }
        # sets avator size, walk movement type, walking speed,
        # and with headlight on
        NavigationInfo
        {
            type "WALK"
            speed 1.0
            headlight TRUE
            avatarSize [ 0.5  1.6  0.5 ]
        }
        # living room with a bar which has four chairs next to it,
        # a sofa, and a tea table
        Inline { url "livingRoom.wrl" }
        # create a door at the right wall that acts as an anchor
        # to the kitchen
```

```
        Anchor
        {
            url "kitchenVP.wrl"
            description "To the kitchen"
            children
            [
                Transform
                {
                    translation  10.25  0.0  -3.0
                    rotation     0.0  1.0  0.0  -1.57
                    children Inline { url "door.wrl" }
                }
            ]
        },
    ]
}


#VRML V2.0 utf8
# file name: myCylinder.wrl
# a transparent cylinder will acts as a cooking pot's
# body
Shape {
      appearance Appearance {
            material Material {
                  diffuseColor 0.0 0.7 1.0
                  transparency 0.5
            }
      }
      geometry Extrusion {
            creaseAngle 0.785
            solid       FALSE
            crossSection [
                   1.00  0.00,   0.92 -0.38,
                   0.71 -0.71,   0.38 -0.92,
                   0.00 -1.00,  -0.38 -0.92,
                  -0.71 -0.71,  -0.92 -0.38,
                  -1.00 -0.00,  -0.92  0.38,
                  -0.71  0.71,  -0.38  0.92,
                   0.00  1.00,   0.38  0.92,
                   0.71  0.71,   0.92  0.38,
                   1.00  0.00
            ]
            spine [ 0.0 -1.0 0.0,  0.0 1.0 0.0 ]
      }
}


#VRML V2.0 utf8
# file name: pot.wrl
# a blue transparent cooking pot that has a knob,
# and two side handles
Group
{
    children
    [
        # builds the pots body with a transparent cylinder
        DEF PotCylinder Inline
        {
            url "myCylinder.wrl"
        },
        # builds a top knob with a sphere
        Transform
        {
```

```
            translation 0.0  1.3  0.0
            children DEF Knob Shape
            {
                appearance Appearance
                {
                    material Material
                    {
                        diffuseColor 0.0  0.7  1.0
                        transparency 0.5
                    }
                }
                geometry Sphere { radius 0.3 }
            }
        },
        # builds a left knob with a scaled sphere
        Transform
        {
            translation -1.31  0.7  0.0
            scale        1.0   0.5  1.0
            children USE Knob
        },
        # builds a left knob with a scaled sphere
        Transform
        {
            translation  1.31  0.7  0.0
            scale        1.0   0.5  1.0
            children USE Knob
        },
    ]
}

#VRML V2.0 utf8
# file name: refrigerator.wrl
# A white refrigerator with a handle
Group
{
    children
    [
        # refrigerator body
        DEF MyBox Shape
        {
            appearance Appearance
            {
                material Material
                {
                    diffuseColor 0.9  0.9  0.9
                }
            }
            geometry Box
            {
                size 2.0  5.0  2.0
            }
        },
        # handle
        Transform
        {
            translation  -0.7  0.6  1.05
            scale         0.1   0.1  0.15
            children USE MyBox
        },
    ]
}
```

```
#VRML V2.0 utf8
# file name: room.wrl
# A room with four brick walls, one of them has a window, a floor,
# and two wall lights
Group
{
    children
    [
        # create a background for this world
        Background
        {
            skyColor
            [
                0.0  0.2  0.7
                0.0  0.5  1.0
                1.0  1.0  1.0
            ]
            skyAngle [ 1.309  1.571 ]
            groundColor
            [
                0.1  0.1  0.0
                0.4  0.25  0.2
                0.6  0.6  0.6
            ]
        }

        # builds a white floor with a flat white box
        Shape
        {
            appearance DEF FloorApp Appearance
            {
                material Material
                {
                    diffuseColor 1.0  1.0  1.0
                }
            }
            geometry DEF FloorGeo Box
            {
                size 21.0  0.2  14.0
            }

        },
        # builds a ceiling with the material and shape as the floor
        # the room gets dark after I put the ceiling on :=(
        #Transform
        #{
        #    translation  0.0  7.0  0.0
        #    children
        #    {
        #        Shape
        #        {
        #            appearance USE FloorApp
        #            geometry USE FloorGeo
        #        }
        #    }
        #},
        # builds a far most wall with three parts,
        # a brick wall with window, and two brick walls on the side
        # 1st part of the far most wall - wall with window
        Transform
        {
            translation 0.0  3.5  -7.0
            children
            [
```

```
            Inline
            {
               url "windowWall.wrl"
            }
        ]
    },
    # 2st part of the far most wall - left to the wall with window
    Transform
    {
        translation -7.0  3.5  -7.0
        children
        [
            DEF BrickWall Inline
            {
               url "wall.wrl"
            }
        ]
    },
    # 3st part of the far most wall - right to the wall with window
    Transform
    {
        translation 7.0  3.5  -7.0
        children USE BrickWall
    },
    # front wall that builds from three parts: front left, front
center, and front right
    # 1st part of font wall - front center wall
    Transform
    {
        translation 0.0  3.5  7.0
        children USE BrickWall
    },
    # 2nd part of font wall - front right wall
    Transform
    {
        translation -7.0  3.5  7.0
        children USE BrickWall
    },
    # 3rd part of font wall - front right wall
    Transform
    {
        translation 7.0  3.5  7.0
        children USE BrickWall
    },
    # builds a left wall with 2 square brick walls
    # far left wall
    Transform
    {
        translation -10.5  3.5  -3.5
        rotation    0.0   1.0  0.0  1.57
        children USE BrickWall
    },
    # front left wall
    Transform
    {
        translation -10.5  3.5  3.5
        rotation    0.0   1.0  0.0  1.57
        children
        [
            USE BrickWall
            DEF Light Inline
            {
               url "light.wrl"
            }
```

```
            ]

        },
        # builds a right wall with 2 square brick walls
        # far right wall
        Transform
        {
            translation 10.5  3.5  -3.5
            rotation    0.0   1.0  0.0  1.57
            children USE BrickWall
        },
        # front right wall
        Transform
        {
            translation 10.5  3.5  3.5
            rotation    0.0   1.0  0.0  1.57
            children
            [
                USE BrickWall
                DEF Light Inline
                {
                   url "light.wrl"
                }
            ]

        },
    ]
}


#VRML V2.0 utf8
# file name: sofa.wrl
# A green sofa built
Group
{
    children
    [
        # sofa seat
        Shape
        {
            appearance DEF Green Appearance
            {
                material Material
                {
                    diffuseColor 0.0 1.0  0.0
                }
            }
            geometry Box
            {
                size 7.0  1.0  3.0
            }
        }
        # sofa back
        Transform
        {
            translation  0.0  0.75  -1.0
            children
            [
                Shape
                {
                    appearance USE Green
                    geometry Box
                    {
                        size 7.0  2.5  1.0
```

```
                    }
                }
            ]
        },

    ]
}


#VRML V2.0 utf8
# file name: table.wrl
# a yellow table that has a table top and four legs
Group
{
    children
    [
        # Table top
        Shape
        {
            appearance DEF Yellow Appearance
            {
                material Material
                {
                    diffuseColor 1.0  1.0  0.0
                }
            }
            geometry Box { size 9  1.0  7 }
        },
        # Left front table leg
        Transform
        {
            translation -3.5  -3.5  2.5
            children DEF TableLeg Shape
            {
                appearance USE Yellow
                geometry Cylinder
                {
                    radius 0.5
                    height 7
                }
            }
        },
        # Right front table leg
        Transform
        {
            translation 3.5  -3.5  2.5
            children USE TableLeg
        },
        # Left rear table leg
        Transform
        {
            translation -3.5  -3.5  -2.5
            children USE TableLeg
        },
        # Left rear table leg
        Transform
        {
            translation 3.5  -3.5  -2.5
            children USE TableLeg
        },
    ]
}
```

```
#VRML V2.0 utf8
# file name: wall.wrl
# Brick wall
Shape {
      appearance Appearance {
            material Material { }
            texture ImageTexture {
                  url "brick.jpg"
            }
            textureTransform TextureTransform {
                  scale 5.0 5.0
            }
      }
      geometry Box { size 7.0 7.0 0.5 }
}


#VRML V2.0 utf8
# file name: window.wrl
# a window built from 4 screens
Group {
      children [
      # Lower-left video screen
            Shape {
                  appearance Appearance {
                        # no material, use emissive texturing
                        texture DEF Video ImageTexture {
                              url "grand.jpg"
                        }
                  }
                  geometry DEF Screen IndexedFaceSet {
                        solid FALSE
                        coord Coordinate {
                              point [
                                    0.0 0.0 0.0,  1.0 0.0 0.0,
                                    1.0 1.0 0.0,  0.0 1.0 0.0,
                              ]
                        }
                        coordIndex [ 0, 1, 2, 3 ]
                        texCoord TextureCoordinate {
                              point [
                                    0.0 0.0,  0.5 0.0,
                                    0.5 0.5,  0.0 0.5,
                              ]
                        }
                        texCoordIndex [ 0, 1, 2, 3 ]
                  }
            },
      # Lower-right video screen
            Transform {
                  translation 1.1 0.0 0.0
                  children Shape {
                        appearance Appearance {
                        # no material, use emissive texturing
                              texture USE Video
                              textureTransform TextureTransform {
                              # Slide to lower-right quadrant
                                    translation 0.5 0.0
                              }
                        }
                        geometry USE Screen
                  }
            },
      # Upper-left video screen
```

```
            Transform {
                  translation 0.0 1.1 0.0
                  children Shape {
                        appearance Appearance {
                        # no material, use emissive texturing
                              texture USE Video
                              textureTransform TextureTransform {
                              # Slide to upper-left quadrant
                                    translation 0.0 0.5
                              }
                        }
                        geometry USE Screen
                  }
            },
      # Upper-right video screen
            Transform {
                  translation 1.1 1.1 0.0
                  children Shape {
                        appearance Appearance {
                        # no material, use emissive texturing
                              texture USE Video
                              textureTransform TextureTransform {
                              # Slide to upper-right quadrant
                                    translation 0.5 0.5
                              }
                        }
                        geometry USE Screen
                  }
            }
      ]
}


#VRML V2.0 utf8
# file name: windowWall.wrl
# Brick wall with a window in the middle
Group
{
    children
    [
        # a brick wall
        Inline
        {
            url "wall.wrl"
        },
        # a white box behind the window acts
        # a window frame
        Transform
        {
            translation 0.0  0.0  0.3
            children
            [
                Shape
                {
                    appearance Appearance
                    {
                        material Material
                        {
                            diffuseColor 1.0  1.0  1.0
                        }
                    }
                    geometry Box { size 2.65 2.65 0.1 }
                }
                # window
```

```
                Transform
                {
                    translation -1.0  -1.0  0.1
                    children
                    [
                        Inline
                        {
                            url "window.wrl"
                        }
                    ]
                }
            ]
        },

    ]
}
```

III. Deliverable 3

“histogram.xml”

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE histogram SYSTEM "histogramDTD.dtd">
<?xml-stylesheet type="text/xsl" href="histogram.xsl"?>
<!-- top element, histograms may contains 0 or more histogram tags  -->
<histograms>
<!-- a histogram may contain 1 or more bins,
     binwidth specifics bin width
     maxNumber specifics the maximum bin height
-->
<histogram binwidth="15" maxNumber="100">
    <!-- number specifics height of each bin -->
    <bin number="100" />
    <bin number="0" />
    <bin number="70" />
    <bin number="88" />
    <bin number="90" />
</histogram>
<histogram binwidth="25" maxNumber="150">
    <bin number="150" />
    <bin number="30" />
    <bin number="0" />
    <bin number="88" />
</histogram>
</histograms>
```

“histogramDTD.dtd”

```
<!-- histograms contains 0 or more histogram -->
<!ELEMENT histograms (histogram) >
<!-- histogram contains 1 or more bins -->
<!ELEMENT histogram (bin+) >
<!-- histogram's binwidth and maxNumber attribute can not be null -->
<!ATTLIST histogram
    binwidth    CDATA   #REQUIRED
    maxNumber   CDATA   #REQUIRED
>
<!-- bin tag contains no data, only attribute number -->
<!ELEMENT bin EMPTY>
<!-- bin tag contains only attribute number -->
<!ATTLIST bin
```

```
    number      CDATA   #REQUIRED
>
```

“histogram.xsl”

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!-- output html, head, and body tag when root element is matched -->
<xsl:template match="/">
    <html xmlns:v="urn:schemas-microsoft-com:vml">
        <head>
            <title>Simple Histogram</title>
            <object id="VMLRender"
                classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
            </object>
            <style>
                v\:* {behavior: url(#VMLRender)}
            </style>
        </head>
        <body>
            <!-- apply histogram templates when reads a histogram tag  -
->
            <xsl:apply-templates/>
        </body>
    </html>
</xsl:template>

<!-- output a table for every histogram tag encountered -->
<xsl:template match="histogram">
    <center>
    <!-- sets table cellspacing and border to 0 -->
    <table cellspacing="0" border="0">
        <tr>
            <!-- output max column number -->
            <th valign="top" align="right" bgcolor="white"><xsl:value-of
select="@maxNumber"/></th>
            <!-- output a bin for each bin tag -->
            <xsl:apply-templates/>
        </tr>
        <tr>
            <!-- displays first table column starts with 0 -->
            <td valign="top" align="right" bgcolor="white">0</td>
            <!-- prints bin's column number -->
            <xsl:for-each select="bin">
            <td><xsl:number level="single" count="bin" format="1"/></td>
            </xsl:for-each>
        </tr>
    </table>
    </center>
</xsl:template>

<!-- output tr tag for displaying bin row -->
<xsl:template match="bin">
    <xsl:element name="td" use-attribute-sets="tdAttributes">
       <!-- output rectangle tag and its attributes for each bin -->
       <!-- output rectangle's open tag, "<" -->
       <xsl:text disable-output-escaping="yes">&lt;</xsl:text>
       <!-- output part of rectangle attribute, 'v:rect style="' -->
       <xsl:text disable-output-escaping="yes">v:rect
style=&quot;</xsl:text>
       <!-- output part of rectangle attribute name height -->
       <xsl:text disable-output-escaping="yes">height:</xsl:text>
       <!-- output rectangle's height -->
```

```
       <xsl:value-of select="@number"/>
       <!-- output rectangle's width attribute name -->
       <xsl:text disable-output-escaping="yes">; width:</xsl:text>
       <!-- output rectangle's width -->
       <xsl:value-of select="../@binwidth"/>
       <!-- set rectangle's top and left position to 0, and end
rectangle tag -->
       <xsl:text disable-output-escaping="yes">; top:0;
left:0&amp;quot;</xsl:text>
       <!-- set bins color to white and close rect tag -->
       <xsl:text disable-output-escaping="yes">filled=&amp;quot;true&amp;quot;
fillcolor=&amp;quot;white&amp;quot; /&amp;gt;</xsl:text>

    </xsl:element>
</xsl:template>

<!-- sets table column attributes for the rectangle that resides -->
<xsl:attribute-set name="tdAttributes">
    <!-- objects in this column will be vertically aligned at the bottom
-->
    <xsl:attribute name="valign">bottom</xsl:attribute>
    <!-- sets table column width to histogram bin width -->
    <xsl:attribute name="width"><xsl:value-of
select="../@binwidth"/></xsl:attribute>
    <!-- sets table column height to histogram max bin number -->
    <xsl:attribute name="height"><xsl:value-of
select="../@maxNumber"/></xsl:attribute>
    <!-- sets table column back ground color to blue -->
    <xsl:attribute name="bgcolor">#0000ff</xsl:attribute>
</xsl:attribute-set>

</xsl:stylesheet>
```

IV. Deliverable 4

“box.xml”

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="box.xsl"?>

<!DOCTYPE X3D SYSTEM "latest.dtd">
<X3D>
      <Scene>
            <Transform>
                  <Shape>
                        <Box size="200.0 200.0 200.0" />
                  </Shape>
            </Transform>
      </Scene>
</X3D>
```

“box.xsl”

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt"
    xmlns:helper="http://deliverable4">

      <msxsl:script language="JavaScript" implements-prefix="helper">
            <![CDATA[
```

```
                var dz = 100;

                /**
                    Returns x coordinate given box's width and index of
this point
                */
                  function xVC(size, index)
                  {
                      // left to the origin
                        if(index == 1 || index == 4 || index == 5 ||
index == 6)
                        {
                              return 0 - size/2;
                        }
                        // right to the origin
                        else if(index == 2 || index == 3 || index == 7
|| index == 8)
                        {
                              return 0 + size/2;
                        }
                  }

                /**
                    Returns y coordinate given box's height and index of
this point
                */
                  function yVC(size, index)
                  {
                      // above the origin
                        if(index == 1 || index == 2 || index == 6 ||
index == 7)
                        {
                              return 0 + size/2;

                        }
                        // under the origin
                        else if(index == 3 || index == 4 || index == 5
|| index == 8)
                        {
                              return 0 - size/2;
                        }
                  }

                  /**
                    Returns z coordinate given box's depth and index of
this point
                */
                  function zVC(size, index)
                  {
                      // infront of the origin
                        if(index == 1 || index == 2 || index == 3 ||
index == 4)
                        {
                              return 0 + size/2 - dz;

                        }
                        // behind the origin
                        else if(index == 5 || index == 6 || index == 7
|| index == 8)
                        {
                              return 0 - size/2 - dz;
                        }
                  }
```

```
                    /*
                        Returns a number's projected value given
                        its view coordinate position and view
                        coordinate position in z-axes
                    */
                    function projectionPos(posVC, zVC)
                    {
                          return (posVC * dz)/(dz - zVC);
                    }

            ]]>

      </msxsl:script>

<!-- output html, head, and body tag when root element is matched -->
<xsl:template match="/">
    <html xmlns:v="urn:schemas-microsoft-com:vml">
        <head>
            <title>Simple Histogram</title>
            <object id="VMLRender"
                classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
            </object>
            <style>
                v\:* {behavior: url(#VMLRender)}
            </style>
        </head>
        <body>
            <!-- apply histogram templates when reads a histogram tag  -
->
            <xsl:apply-templates/>
        </body>
    </html>
</xsl:template>

<xsl:template match="/">
            <xsl:apply-templates />
      </xsl:template>
      <xsl:template match="X3D">
            <!--<xsl:text disable-output-escaping="yes">-->
                        <html xmlns:v="urn:schemas-microsoft-com:vml">
                    <head>
                        <title>Box tag from X3D to VML</title>
                            <object id="VMLRender"
                                classid="CLSID:10072CEC-8CC1-11D1-986E-
00A0C955B42E">
                            </object>
                    <style>
                        v\:* {behavior: url(#VMLRender)}
                    </style>
                </head>
                <body>
                    <br/>
                    <br/>
                    <br/>
                    <br/>
                    <br/>
                    <center>
                    <!-- apply templates when reads a tag  -->
                    <xsl:apply-templates/>
                    </center>
                </body>
            </html>

            <!--</xsl:text>-->
```

```
	</xsl:template>
	<xsl:template match="Scene">
		<xsl:apply-templates />
	</xsl:template>
	<xsl:template match="Transform">
		<xsl:apply-templates />
	</xsl:template>
	<xsl:template match="Shape">
		<xsl:apply-templates />
	</xsl:template>
	<xsl:template match="Box">
		<!--In VRML, all primitive shapes are create at original
position.-->
		<!--In this translation, we will move object to the z-
negative position by 100 units -->
		<!--We specify a veiw plane is XY plane at z = 100, and the
center reference point-->
		<!--for projection is (0, 0, 100)-->

		<!-- gets box's sizes from box tag -->
		<xsl:variable name="boxDim" select="@size"/>
		<xsl:variable name="x" select="substring-before($boxDim, '
')"/>
		<xsl:variable name="rest" select="substring-after($boxDim, '
')"/>
		<xsl:variable name="y" select="substring-before($rest, '
')"/>
		<xsl:variable name="z" select="substring-after($rest, '
')"/>

		<xsl:variable name="xVC1" select="helper:xVC($x, 1)" />
		<xsl:variable name="yVC1" select="helper:yVC($y, 1)" />
		<xsl:variable name="zVC1" select="helper:zVC($z, 1)" />
		<xsl:variable name="xVC2" select="helper:xVC($x, 2)" />
		<xsl:variable name="yVC2" select="helper:yVC($y, 2)" />
		<xsl:variable name="zVC2" select="helper:zVC($z, 2)" />
		<xsl:variable name="xVC3" select="helper:xVC($x, 3)" />
		<xsl:variable name="yVC3" select="helper:yVC($y, 3)" />
		<xsl:variable name="zVC3" select="helper:zVC($z, 3)" />
		<xsl:variable name="xVC4" select="helper:xVC($x, 4)" />
		<xsl:variable name="yVC4" select="helper:yVC($y, 4)" />
		<xsl:variable name="zVC4" select="helper:zVC($z, 4)" />
		<xsl:variable name="xVC5" select="helper:xVC($x, 5)" />
		<xsl:variable name="yVC5" select="helper:yVC($y, 5)" />
		<xsl:variable name="zVC5" select="helper:zVC($z, 5)" />
		<xsl:variable name="xVC6" select="helper:xVC($x, 6)" />
		<xsl:variable name="yVC6" select="helper:yVC($y, 6)" />
		<xsl:variable name="zVC6" select="helper:zVC($z, 6)" />
		<xsl:variable name="xVC7" select="helper:xVC($x, 7)" />
		<xsl:variable name="yVC7" select="helper:yVC($y, 7)" />
		<xsl:variable name="zVC7" select="helper:zVC($z, 7)" />
		<xsl:variable name="xVC8" select="helper:xVC($x, 8)" />
		<xsl:variable name="yVC8" select="helper:yVC($y, 8)" />
		<xsl:variable name="zVC8" select="helper:zVC($z, 8)" />
		<xsl:variable name="xP1" select="helper:projectionPos($xVC1,
$zVC1)" />
		<xsl:variable name="yP1" select="helper:projectionPos($yVC1,
$zVC1)" />
		<xsl:variable name="xP2" select="helper:projectionPos($xVC2,
$zVC2)" />
		<xsl:variable name="yP2" select="helper:projectionPos($yVC2,
$zVC2)" />
		<xsl:variable name="xP3" select="helper:projectionPos($xVC3,
$zVC3)" />
```

```
            <xsl:variable name="yP3" select="helper:projectionPos($yVC3,
$zVC3)" />
            <xsl:variable name="xP4" select="helper:projectionPos($xVC4,
$zVC4)" />
            <xsl:variable name="yP4" select="helper:projectionPos($yVC4,
$zVC4)" />
            <xsl:variable name="xP5" select="helper:projectionPos($xVC5,
$zVC5)" />
            <xsl:variable name="yP5" select="helper:projectionPos($yVC5,
$zVC5)" />
            <xsl:variable name="xP6" select="helper:projectionPos($xVC6,
$zVC6)" />
            <xsl:variable name="yP6" select="helper:projectionPos($yVC6,
$zVC6)" />
            <xsl:variable name="xP7" select="helper:projectionPos($xVC7,
$zVC7)" />
            <xsl:variable name="yP7" select="helper:projectionPos($yVC7,
$zVC7)" />
            <xsl:variable name="xP8" select="helper:projectionPos($xVC8,
$zVC8)" />
            <xsl:variable name="yP8" select="helper:projectionPos($yVC8,
$zVC8)" />


         <!--link point 1 and 2 -->
        <!-- output line tag and its attributes for each line -->
        <!-- output line's open tag, "<" -->
        <xsl:text disable-output-escaping="yes">&lt;</xsl:text>
        <!-- output part of line attribute, 'v:line' -->
        <xsl:text disable-output-escaping="yes">v:line </xsl:text>
        <!-- output part of line attribute from -->
        <xsl:text disable-output-escaping="yes">from="</xsl:text>
        <!-- output rectangle's from attribute for x and y -->
        <xsl:value-of select="$xP1"/>
        <xsl:text disable-output-escaping="no"> </xsl:text>
        <xsl:value-of select="$yP1"/>
        <!-- output rectangle's to attribute for x and y -->
        <xsl:text disable-output-escaping="yes">" to="</xsl:text>
        <!-- output rectangle's width -->
        <xsl:value-of select="$xP2"/>
        <xsl:text disable-output-escaping="no"> </xsl:text>
        <xsl:value-of select="$yP2"/>
        <!-- close line tag -->
        <xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


            <!--link point 2 and 3 -->
            <!-- output line tag and its attributes for each line -->
        <!-- output line's open tag, "<" -->
        <xsl:text disable-output-escaping="yes">&lt;</xsl:text>
        <!-- output part of line attribute, 'v:line' -->
        <xsl:text disable-output-escaping="yes">v:line </xsl:text>
        <!-- output part of line attribute from -->
        <xsl:text disable-output-escaping="yes">from="</xsl:text>
        <!-- output rectangle's from attribute for x and y -->
        <xsl:value-of select="$xP2"/>
        <xsl:text disable-output-escaping="no"> </xsl:text>
        <xsl:value-of select="$yP2"/>
        <!-- output rectangle's to attribute for x and y -->
        <xsl:text disable-output-escaping="yes">" to="</xsl:text>
        <!-- output rectangle's width -->
        <xsl:value-of select="$xP3"/>
        <xsl:text disable-output-escaping="no"> </xsl:text>
        <xsl:value-of select="$yP3"/>
```

```
<!-- close line tag -->
<xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


     <!--link point 3 and 4 -->
     <!-- output line tag and its attributes for each line -->
<!-- output line's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of line attribute, 'v:line' -->
<xsl:text disable-output-escaping="yes">v:line </xsl:text>
<!-- output part of line attribute from -->
<xsl:text disable-output-escaping="yes">from="</xsl:text>
<!-- output rectangle's from attribute for x and y -->
<xsl:value-of select="$xP3"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP3"/>
<!-- output rectangle's to attribute for x and y -->
<xsl:text disable-output-escaping="yes">" to="</xsl:text>
<!-- output rectangle's width -->
<xsl:value-of select="$xP4"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP4"/>
<!-- close line tag -->
<xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


     <!--link point 4 and 5 -->
     <!-- output line tag and its attributes for each line -->
<!-- output line's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of line attribute, 'v:line' -->
<xsl:text disable-output-escaping="yes">v:line </xsl:text>
<!-- output part of line attribute from -->
<xsl:text disable-output-escaping="yes">from="</xsl:text>
<!-- output rectangle's from attribute for x and y -->
<xsl:value-of select="$xP4"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP4"/>
<!-- output rectangle's to attribute for x and y -->
<xsl:text disable-output-escaping="yes">" to="</xsl:text>
<!-- output rectangle's width -->
<xsl:value-of select="$xP5"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP5"/>
<!-- close line tag -->
<xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


     <!--link point 5 and 6 -->
     <!-- output line tag and its attributes for each line -->
<!-- output line's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of line attribute, 'v:line' -->
<xsl:text disable-output-escaping="yes">v:line </xsl:text>
<!-- output part of line attribute from -->
<xsl:text disable-output-escaping="yes">from="</xsl:text>
<!-- output rectangle's from attribute for x and y -->
<xsl:value-of select="$xP5"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP5"/>
<!-- output rectangle's to attribute for x and y -->
<xsl:text disable-output-escaping="yes">" to="</xsl:text>
<!-- output rectangle's width -->
<xsl:value-of select="$xP6"/>
```

```
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP6"/>
<!-- close line tag -->
<xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


     <!--link point 6 and 7 -->
     <!-- output line tag and its attributes for each line -->
<!-- output line's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of line attribute, 'v:line' -->
<xsl:text disable-output-escaping="yes">v:line </xsl:text>
<!-- output part of line attribute from -->
<xsl:text disable-output-escaping="yes">from="</xsl:text>
<!-- output rectangle's from attribute for x and y -->
<xsl:value-of select="$xP6"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP6"/>
<!-- output rectangle's to attribute for x and y -->
<xsl:text disable-output-escaping="yes">" to="</xsl:text>
<!-- output rectangle's width -->
<xsl:value-of select="$xP7"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP7"/>
<!-- close line tag -->
<xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


     <!--link point 7 and 8 -->
     <!-- output line tag and its attributes for each line -->
<!-- output line's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of line attribute, 'v:line' -->
<xsl:text disable-output-escaping="yes">v:line </xsl:text>
<!-- output part of line attribute from -->
<xsl:text disable-output-escaping="yes">from="</xsl:text>
<!-- output rectangle's from attribute for x and y -->
<xsl:value-of select="$xP7"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP7"/>
<!-- output rectangle's to attribute for x and y -->
<xsl:text disable-output-escaping="yes">" to="</xsl:text>
<!-- output rectangle's width -->
<xsl:value-of select="$xP8"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP8"/>
<!-- close line tag -->
<xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


     <!--link point 1 and 4 -->
     <!-- output line tag and its attributes for each line -->
<!-- output line's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of line attribute, 'v:line' -->
<xsl:text disable-output-escaping="yes">v:line </xsl:text>
<!-- output part of line attribute from -->
<xsl:text disable-output-escaping="yes">from="</xsl:text>
<!-- output rectangle's from attribute for x and y -->
<xsl:value-of select="$xP1"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP1"/>
<!-- output rectangle's to attribute for x and y -->
<xsl:text disable-output-escaping="yes">" to="</xsl:text>
```

```
<!-- output rectangle's width -->
<xsl:value-of select="$xP4"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP4"/>
<!-- close line tag -->
<xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


     <!--link point 5 and 8 -->
     <!-- output line tag and its attributes for each line -->
<!-- output line's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of line attribute, 'v:line' -->
<xsl:text disable-output-escaping="yes">v:line </xsl:text>
<!-- output part of line attribute from -->
<xsl:text disable-output-escaping="yes">from="</xsl:text>
<!-- output rectangle's from attribute for x and y -->
<xsl:value-of select="$xP5"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP5"/>
<!-- output rectangle's to attribute for x and y -->
<xsl:text disable-output-escaping="yes">" to="</xsl:text>
<!-- output rectangle's width -->
<xsl:value-of select="$xP8"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP8"/>
<!-- close line tag -->
<xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


     <!--link point 1 and 6 -->
     <!-- output line tag and its attributes for each line -->
<!-- output line's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of line attribute, 'v:line' -->
<xsl:text disable-output-escaping="yes">v:line </xsl:text>
<!-- output part of line attribute from -->
<xsl:text disable-output-escaping="yes">from="</xsl:text>
<!-- output rectangle's from attribute for x and y -->
<xsl:value-of select="$xP1"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP1"/>
<!-- output rectangle's to attribute for x and y -->
<xsl:text disable-output-escaping="yes">" to="</xsl:text>
<!-- output rectangle's width -->
<xsl:value-of select="$xP6"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP6"/>
<!-- close line tag -->
<xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


     <!--link point 2 and 7 -->
     <!-- output line tag and its attributes for each line -->
<!-- output line's open tag, "<" -->
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
<!-- output part of line attribute, 'v:line' -->
<xsl:text disable-output-escaping="yes">v:line </xsl:text>
<!-- output part of line attribute from -->
<xsl:text disable-output-escaping="yes">from="</xsl:text>
<!-- output rectangle's from attribute for x and y -->
<xsl:value-of select="$xP2"/>
<xsl:text disable-output-escaping="no"> </xsl:text>
<xsl:value-of select="$yP2"/>
```

```
        <!-- output rectangle's to attribute for x and y -->
        <xsl:text disable-output-escaping="yes">" to="</xsl:text>
        <!-- output rectangle's width -->
        <xsl:value-of select="$xP7"/>
        <xsl:text disable-output-escaping="no"> </xsl:text>
        <xsl:value-of select="$yP7"/>
        <!-- close line tag -->
        <xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


             <!--link point 3 and 8 -->
             <!-- output line tag and its attributes for each line -->
        <!-- output line's open tag, "<" -->
        <xsl:text disable-output-escaping="yes">&lt;</xsl:text>
        <!-- output part of line attribute, 'v:line' -->
        <xsl:text disable-output-escaping="yes">v:line </xsl:text>
        <!-- output part of line attribute from -->
        <xsl:text disable-output-escaping="yes">from="</xsl:text>
        <!-- output rectangle's from attribute for x and y -->
        <xsl:value-of select="$xP3"/>
        <xsl:text disable-output-escaping="no"> </xsl:text>
        <xsl:value-of select="$yP3"/>
        <!-- output rectangle's to attribute for x and y -->
        <xsl:text disable-output-escaping="yes">" to="</xsl:text>
        <!-- output rectangle's width -->
        <xsl:value-of select="$xP8"/>
        <xsl:text disable-output-escaping="no"> </xsl:text>
        <xsl:value-of select="$yP8"/>
        <!-- close line tag -->
        <xsl:text disable-output-escaping="yes">"/&gt;</xsl:text>


      </xsl:template>

</xsl:stylesheet>
```

“latest.dtd” (this DTD is downloaded from the below URL)

```
<!--
# "http://www.web3D.org/TaskGroups/x3d/sun/latest.dtd"
# "http://www.web3D.org/TaskGroups/x3d/sun/latest.html"
# this X3D DTD:
#     http://www.web3D.org/TaskGroups/x3d/sun/latest.dtd
#
# test world:
#     http://www.web3D.org/TaskGroups/x3d/translation/AllVrml97Nodes.wrl
#
      http://www.web3D.org/TaskGroups/x3d/translation/AllVrml97Nodes.wrl
.txt
# test translation:
#     http://www.web3D.org/TaskGroups/x3d/translation/AllVrml97Nodes.xml
#
      http://www.web3D.org/TaskGroups/x3d/translation/AllVrml97Nodes.xml
.txt
#
# DTD version plans:
#     increment by .1 when each of the following is complete:
#     - initial well-defined DTD                                    0.5
#     - all nodes available in test scene and checked in DTD        0.6
#     - Xeena able to present translated X3D scene using DTD        0.7
#     - Xj3D able to translate test scene using DTD into X3D        0.8
#     - DTD compliance exercised by NIST's VRML & XML test suites   0.9
#     - approval by task group, community, Web3D Consortium         1.0
```

```
#       (estimated February 2000)
#
# current version:      0.7999
#
# dtd update activity:
#
# rmgold 12/2/99
#     merged and verified Don's update vs. xeena and test object
           ( make it 0.8 when ready )
#     cleaned up Protos, ProtoDefines, is maps, external Protos
#     removed eventIn/Out/exposed's
#     added oldFieldTypes
#     recommented datatypes [] ()
#     removed the Box field was causing parse error
#     need to fix script better
#
# brutzman 12/1/99
#     Coordinate's 'point' attribute and Color's 'color attribute
changed
#       from #REQUIRED to #IMPLIED (i.e. such values are optional, not
#       required) in order to match VRML 97 spec.  Another way to look
at
#       this change:   NULL values are now permitted.
#     Removed [square brackets] and (parentheses) from default values to
#       match earlier decision to remove them from numeric fields.
# brutzman 11/27/99
#     Produced a number of fixes in Xj3D source:  cvs and build script
work
#     Xeena able to present translated X3D scene using DTD without
problems
#     DTD unchanged, upgraded to version 0.7
# brutzman 11/13/99
#     Verified a number of fixes accomplished in Xj3D, DTD unchanged
# brutzman 11/9/99
#     Sound:  simplified definition to ( AudioClip | MovieTexture )
#       since no more than one (SFNode) is allowed
# brutzman 11/4/99
#     Testing with updated Xj3D - thanks Rick!
#     ROUTE changed to Route for translatability with current build,
#       any opinions?  ROUTE and various *PROTO* tags might be
capitalized
#       to match VRML 97 or left single-cap only to match other tags
#     Route:  from, to, sourceEvent, targetEvent -> fromNode, fromField,
#       toNode, toField.  Much clearer name suggestions from Xj3D.  Not
a
#       problem, VRML 97 specification does not specify strict names for
#       these terms.
#
# brutzman 11/3/99
#     Collision:  can contain at most one Proxy, in any sequence with
#       other ChildrenNodes.
# brutzman 11/2/99
#     PROTO-instance:  for overriding field defaults, now trying tag
#       fieldReset -> defaultValue
#     FieldTypes naming:  Vector3s -> Vector3Array, Vector4s ->
Vector4Array
#     field, exposedField, defaultValue:  made 'value' a REQUIRED
attribute
#     eventIn value defaults to "" (just like eventOut value)
# brutzman 11/1/99
#     IS:  interface -> interfaceFieldName, internal ->
internalFieldName
#     CylinderSensor:  trackpoint -> trackPoint to match VRML 97
eventOut
```

```
#     TouchSensor:  added hitTexCoord to match VRML 97 eventOut
#     Renamed the PROTO definition tag  PROTO -> PROTO-declare, renamed
#       the PROTO instantiation tag PROTO-use -> PROTO-instance, renamed
#       EXTERNPROTO instantiation tag EXTERNPROTO -> EXTERNPROTO-
declare.
#       Goal is clarity during development.  Note completely independent
#       semantics from DEF/USE, thus just trying different tag names.
#       PROTO-declare can contain
eventIn/eventOut/field/exposedField/IS/nodes,
#       EXTERNPROTO-declare is url plus
eventIn/eventOut/field/exposedField.
#       PROTO-instance can only contain reinitialized field/exposedField
tags.
#       PROTO-instance 'type' attribute renamed 'name' to match PROTO-
declare.
#       PROTO-instance contains fieldReset instead of field/exposedField
#         so that individual field types don't have to be redeclared.
#       Considering single unnested ID/IDREF namespace to eliminate
ambiguity.
#       Stay tuned, this whole area related to PROTOs is still in flux.
# brutzman 10/31/99
#     PROTO:  added exposedField, 'type' attribute renamed 'name', added
IS,
#       ProtoUse renamed PROTO-use
#     initial version of EXTERNPROTO added, looks pretty simple actually
#     <field>, <eventIn>, <eventOut>:  attribute 'id' renamed to 'name'
#     added <exposedField>, value required for field and exposedField
#     added IS inside PROTO, with attributes 'interface' and 'internal'
#     Open issue:  how do we include and check for named
PROTO/EXTERNPROTO nodes
#       when ID/IDREF limited to a single namespace
#     Inline:  allowed to contain SceneNodes (as demonstrated by Xj3D)
in order to permit
#       Inline'd content in the scene graph.  Note this allows storing
intermediate results.
#     Type names:  changed Bool|Bools -> Boolean|Booleans, deleted
Norm|Norms,
#       changed Vec|Vecs -> Vector3|Vector3s|Vector4|Vector4s,
#       added Image, Rotation|Rotations, String|Strings, Time|Times.
#       Do people like these names better than SFVec3f|MFVec3f etc?
#     Finished initial testing of all VRML 97 nodes using DTD and
Xj3D/Xeena
#     Keeping Don's testing build updated with Rick's bugfixes and CVS
still a challenge
# brutzman 10/29/99
#     Appearances:  partially relaxed order, either Material or one of
TextureNodes first
#     Various fields, especially eventOuts:  relaxed default value
constraint from FIXED
#       (i.e. only one valid constant value) to default value (any value
OK, default
#       provided if none given).  This permits validly
saving/reloading/validating state,
#       even of prior eventOut values.  Also eliminates perhaps-
unnecessary warnings
#       if non-default values are ever used.
#       ** Need independent confirmation this makes sense, please!
#     IndexedFaceSet: partially relaxed order, either Color or
Coordinate first
#       (as long as they precede Normal?, TextureCoordinate?)
# brutzman 10/20/99
#     WorldInfo:  added to &SceneLeafNodes; since valid as a child node.
appropriate
```

```
#       to also kept it as one of &SceneNodes; since no apparent
repetition collision.
#     %BindableNodes; added to &ChildrenNodes;
#     Alphabetized %BehaviorLeafNodes;
#     Switch:  changed from ( %ChildrenNodes; )* to %Children; to match
other
#       grouping nodes (no semantic change)
#     Better alphabetized order of various ParameterEntities
# brutzman 10/19/99
#     Switch:  changed content from &GroupNodes; to %ChildrenNodes;
#       (but now note %ChildrenNodes; doesn't match "children nodes" in
#       "VRML 97 4.6.5 Grouping and children nodes")
# sonstein 10/18/99
#     fix PROTO's capitalization of (eventIn | eventOut | field)+
#     remove 'byteCodes' from %BehaviorLeafNodes
# brutzman 10/18/99
#     IndexedFaceSet, IndexedLineSet:  allowed zero or one Coordinate
node
#       (instead of mandatory one) since coord NULL is legal
#     IndexedLineSet:  allowed Color, Coordinate in either order
#     Interpolators:  fraction and value fields consistently initialized
#     Pointset:  corrected definition, added DEF/USE
#     Switch:  changed content from Group to %GroupNodes;
#     Switch:  allow multiple group nodes under 'choice'
# brutzman 10/17/99
#     Are special conversion rules needed to handle embedded blanks in
URL fields?
#     NavigationInfo's type attribute too restrictive using ( ANY | ...
) selections,
#       needs to support zero-to-several of at least the following
navigation types:
#       "ANY", "WALK", "EXAMINE", "FLY", and "NONE"
#       so relaxed attribute constraint to NMTOKENS #IMPLIED (default
value implied)
#       and translations should not include [ brackets around type
values ]
#     Added consistent bind, bindTime, isBound events to Background,
Fog, NavigationInfo
#       and Viewpoint (the bindable nodes).  Note VRML 97 spec
inconsistent about these.
#       Default binding values false, since browser/user decides.
# brutzman 10/16/99
#     Changed DEF/USE back to ID/IDREF, translation issues posted on
list
#       http://www.web3d.org/WorkingGroups/x3d-
contributors/hypermail/1999/1713.html
#       http://www.web3d.org/WorkingGroups/x3d-
contributors/hypermail/1999/1719.html
#     Added examine field to Viewpoint (at least temporarily while being
discussed)
#       http://www.web3d.org/WorkingGroups/x3d-
contributors/hypermail/1999/1720.html
#     Auvo's redefined Shape permits arbitrary order for Appearance &
GeometryNodes
# brutzman 10/14/99
#     Need to look at defaults for all eventIns & eventOuts
#     looking at using NMTOKEN instead of ID/IDREF for DEF/USE values -
#       permits name repetition (like VRML 97), still enforces legal
name strings
# brutzman 10/13/99
#     Added IndexedLineSet
#     Renamed Normal's normal field to vector to match VRML 97
#     Renamed TextureCoordinate's texCoord field to point to match VRML
97
```

```
#     PointSet:  changed order to ( ColorSet?, CoordinateSet) to match
VRML 97
#     changed various interpolator eventIn fraction fields to #FIXED
#     to do:  ensure all events added, with attribute default #FIXED.
defaults?
#     added Script - check user-defined tags eventIn, field, eventOut
#     decapitalized eventIn, field, eventOut to match VRML 97
#     problem:  when USEing another node, the DEF'ed name is erroneously
duplicated
#     Sound:  added MovieTexture as possible source
# brutzman 10/12/99
#     Updated lots of cosmetic spacing
#     Added colorPerVertex to IndexedFaceSet
# brutzman 10/11/99
#     Added PixelTexture to TextureNodes, created PixelTexture element
#     modified Shape to take only zero or one of GeometryNodes
#     alphabetized GeometryNodes to match VRML 97 spec
# brutzman 10/9/99
#     added optional Header tag & kept Scene tag based on Auvo's
suggestions
#     removed encoding attribute from X3D & Scene tags, since it is in
xml 1.0 header
#     decided to enter attributes for
field/exposedField/eventIn/eventOut 's in VRML
#     Background:  added bottomUrl, groundColor value "(0 0 0)",
bindTime, isBound
# brutzman 10/8/99
#     added bboxCenter, bboxCenter to Anchor, Billboard, Collision,
Group,
#       Inline, Transform
#     reordered fields and eliminated some .0's to match VRML 97
specification
#     Need to add <Proxy /> under Collision,
#       added ProxyNodes parameter entity (since can't use Collision
under proxy)
#       split up Children into ChildrenNodes and Children to help Proxy
definitions
#       need to verify it works and nests...
#     Changed content particle repetition operator (from + to *) for
#       parameter entity & Children, thus permitting absence of child
nodes
#     Alphabetized nodes
#     Added X3D node (e.g. HTML spec), apparently can eliminate Scene
now
#     Changed Anchor URL from REQUIRED to IMPLIED since VRML 97 permits
empty URL
#     Changed TRUE to true, FALSE to false, matching DOM and Xj3D
# v0.2.5
# rmgold 8/03/99
#     set bindables bind to default TRUE since that works properly now
# rmgold 7/23/99
#     moved Coordinate,Normal,Color,TextureCoordinate data
#        from PCDATA to an attribute CDATA
#     moved Index to CDATA attribute of IndexedFaceSet
# rmgold 7/19/99
#     reverted Text
# rmgold 7/16/99
#     collapsed Geometry tag
# rmgold 6/30/99                cleaned up Proto section
# rmgold 5/25/99
#     comment the html generator and add link to parser, more comment
cleaning
# brutzman 5/24/99
```

```
#     DTD naming convention:  latest.dtd & latest.html, older versions
dated
# rmgold 5/24/99
#     followed more ENTITY defs
# brutzman, glidden 5/21/99   defined parameter entities for GroupNodes
and BindableNodes
#     (used by Scene and Proto), named dtd, capitalized DEF and USE,
#     renamed Proto, ProtoDef, Route to PROTO, PROTObody and ROUTE;
#     changed PROTO "type" to "name"
#     expect that PROTOs really need to be internal entities (i.e. DTD
fragments)
#         in x3dfile.xml scenes, need to scrub VRML primitive type
enumerations
# rmgold 5/20/99
#     added ENTITY DEFs for Children, FieldTypes
      renamed ProtoInstance to X3D (0.2.3)
# rmgold 5/13/99
#     reintroduced Proto, there is a minimally non validating
      way to do this
# rmgold 5/12/99
#     removed Interface def, removed Proto there is no clean
      way to do this
# rmgold 5/12/99
#     moved around Proto, new ProtoDef, DEF/USE ( 0.2.1 )
      fixed CoordinateSet and cousins to use PCDATA
      fixed ByteCode to use Interface for consistency
      with Proto
      fixed missing Bool | Bools field types
# rmgold 5/11/99
#     fixed various vrml abuse ( Geometries directly in Group for
example )
#     renamed root node to "scene" ( 0.1.2 )
#     added Proto and Interface def'ns
# brutzman 5/10/99
#     fixed lots of syntax abuse - now xeena will parse no problem (
0.1.1 )
# rmgold 5/6/99
#     fixed axisOfRotation field in Billboard
      fixed group types in toplevel scene
      fixed Index attlist
# rmgold 5/5/99
#     completed most nodes ( version 0.1 )
-->


<!-- CDATA field types fall into the following ( ed. lets remove [] and
()? )
     type   example
     Bool   "true"
     Bools  "true false ...  true"
     Vec    "a b c"
     Float  "x.0"
     Floats "a.x b.y ... z.z "
     Color  "r g b" where r,g,b 0.0=>x=>1.0
     Colors
     Norm   "i j k" where sqrt(i*i + j*j + k*k) == 1.0
     Norms
     Int    "I"
     Vecs   "a b c, d e f, ... , g h i "
     Ints   "A B C ... D"
     Rotation
     Rotations

See sample parser generator written using JavaCC at:
```

```
http://www.web3D.org/TaskGroups/x3d/sun/X3DFieldParser.html

-->

<!ENTITY % FieldTypes "( Boolean | Booleans | Color | Colors | Float |
Floats | Int | Ints | Rotation | Rotations | String | Strings | Time |
Times | Tripplet | TrippletArray | Doublet | DoubletArray )" >

<!ENTITY % OldFieldTypes "( SFBool | MFBool | SFColor | MFColor |
SFFloat | MFFloat | SFInt32 | MFInt32 | SFRotation | MFRotation |
SFString | MFString | SFTime | MFTime | SFVec3f | MFVec3f | SFVec2f |
MFVec2f )" >

<!ENTITY % BehaviorLeafNodes   " ColorInterpolator |
CoordinateInterpolator | CylinderSensor | NormalInterpolator |
OrientationInterpolator | PlaneSensor | PositionInterpolator |
ProximitySensor | ScalarInterpolator | Script | SphereSensor |
TimeSensor | TouchSensor " >

<!ENTITY % BindableNodes       " Background | Fog | NavigationInfo |
Viewpoint " >

<!ENTITY % CollisionProxyNodes " Anchor | Billboard | Group | Inline |
LOD | Switch | Transform " >

<!ENTITY % GroupingNodes       " Anchor | Billboard | Collision | Group
| Inline | LOD | Switch | Transform " >

<!ENTITY % GeometryNodes       " ( Box | Cone | Cylinder | ElevationGrid
| Extrusion | IndexedFaceSet | IndexedLineSet | PointSet | Sphere | Text
) " >

<!ENTITY % LightNodes          " DirectionalLight | SpotLight |
PointLight " >

<!ENTITY % TextureNodes        " ImageTexture | MovieTexture |
PixelTexture " >


<!ENTITY % Appearances         " ( ( Material?, (%TextureNodes;)?,
TextureTransform? ) | ( (%TextureNodes;)?, Material?, TextureTransform?
) ) " >

<!ENTITY % SceneLeafNodes      " Shape | Sound | WorldInfo |
%LightNodes; | Proto " >

<!ENTITY % SceneNodes          " (ProtoDefine)*, ( %GroupingNodes; |
%BindableNodes; | %BehaviorLeafNodes; | %LightNodes; | WorldInfo )*,
Route* " >

<!ENTITY % ChildrenNodes       " %BehaviorLeafNodes; | %BindableNodes; |
%GroupingNodes; | %SceneLeafNodes; " >

<!ENTITY % Children            "( %ChildrenNodes; )* " >



<!ELEMENT X3D    ( Header? , Scene ) >

<!ELEMENT Header ( #PCDATA ) >
<!ATTLIST Header
      status CDATA "reserved for future use" >
```

```
<!ELEMENT Scene ( %SceneNodes; ) >

<!ELEMENT Anchor %Children; >
<!ATTLIST Anchor
      description CDATA #IMPLIED
      url         CDATA #IMPLIED
      parameter   CDATA #IMPLIED
      bboxCenter  CDATA "0 0 0"
      bboxSize    CDATA "-1 -1 -1"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED >

<!ELEMENT Appearance ( %Appearances; ) >
<!ATTLIST Appearance
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT AudioClip EMPTY >
<!ATTLIST AudioClip
      description CDATA #IMPLIED
      loop      (true|false) "false"
      pitch       CDATA "1.0"
      startTime   CDATA "0"
      stopTime    CDATA "0"
      url         CDATA #IMPLIED
      duration    CDATA "0.0"
      isActive  (true|false) "false"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT Background EMPTY>
<!ATTLIST Background
      groundAngle CDATA #IMPLIED
      groundColor CDATA "0 0 0"
      backUrl     CDATA #IMPLIED
      bottomUrl   CDATA #IMPLIED
      frontUrl    CDATA #IMPLIED
      leftUrl     CDATA #IMPLIED
      rightUrl    CDATA #IMPLIED
      topUrl      CDATA #IMPLIED
      skyAngle    CDATA #IMPLIED
      skyColor    CDATA "0 0 0"
      bind        (true|false) "false"
      bindTime    CDATA "-1"
      isBound     (true|false) "false"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED >

<!ELEMENT Billboard %Children; >
<!ATTLIST Billboard
      axisOfRotation CDATA "0 1 0"
      bboxCenter     CDATA "0 0 0"
      bboxSize       CDATA "-1 -1 -1"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT Box EMPTY>
<!ATTLIST Box
      size CDATA "2 2 2"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT Collision ( ( %ChildrenNodes; )*, Proxy?, ( %ChildrenNodes; )*
)  >
```

```
<!ATTLIST Collision
      collide (true|false) "true"
      bboxCenter  CDATA "0 0 0"
      bboxSize    CDATA "-1 -1 -1"
      collideTime CDATA "0.0"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED >

<!ELEMENT Color EMPTY >
<!ATTLIST Color
      color CDATA #IMPLIED
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT ColorInterpolator EMPTY>
<!ATTLIST ColorInterpolator
      key      CDATA #IMPLIED
      keyValue CDATA #IMPLIED
      fraction CDATA "0"
      value    CDATA "0 0 0"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT Cone EMPTY>
<!ATTLIST Cone
      bottomRadius CDATA "1"
      height       CDATA "2"
      side   (true|false) "true"
      bottom (true|false) "true"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT Coordinate EMPTY >
<!ATTLIST Coordinate
      point CDATA #IMPLIED
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT CoordinateInterpolator EMPTY>
<!ATTLIST CoordinateInterpolator
      key      CDATA #IMPLIED
      keyValue CDATA #IMPLIED
      fraction CDATA "0"
      value    CDATA "0 0 0"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT Cylinder EMPTY>
<!ATTLIST Cylinder
      bottom (true|false) "true"
      height CDATA "2"
      radius CDATA "1"
      side   (true|false) "true"
      top    (true|false) "true"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT CylinderSensor EMPTY>
<!ATTLIST CylinderSensor
      autoOffset (true|false) "true"
      diskAngle  CDATA "0.262"
      enabled    (true|false) "true"
      maxAngle   CDATA "-1"
      minAngle   CDATA "0"
```

```
      offset     CDATA "0"
      isActive   (true|false) "false"
      rotation   CDATA "0 0 1 0"
      trackPoint CDATA "0 0 0"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT DirectionalLight EMPTY>
<!ATTLIST DirectionalLight
      ambientIntensity CDATA "0"
      color            CDATA "1 1 1"
      direction        CDATA "0 0 -1"
      intensity        CDATA "1"
      on               (true|false) "true"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT ElevationGrid ( (Color?), (Normal?), (TextureCoordinate?)) >
<!ATTLIST ElevationGrid
      height          CDATA #IMPLIED
      ccw             (true|false) "true"
      colorPerVertex  (true|false) "true"
      creaseAngle     CDATA "0"
      normalPerVertex (true|false) "true"
      solid           (true|false) "true"
      xDimension      CDATA "0"
      xSpacing        CDATA "1.0"
      zDimension      CDATA "0"
      zSpacing        CDATA "1.0"
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED>

<!ELEMENT Extrusion EMPTY >
<!ATTLIST Extrusion
      beginCap     (true|false) "true"
      ccw          (true|false) "true"
      convex       (true|false) "true"
      creaseAngle  CDATA "0.0"
      crossSection CDATA "1 1, 1 -1, -1 -1, -1 1, 1 1"
      endCap       (true|false) "true"
      orientation  CDATA "0 0 1 0"
      scale        CDATA "1 1"
      solid        (true|false) "true"
      spine        CDATA "0 0 0, 0 1 0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Fog EMPTY >
<!ATTLIST Fog
      color           CDATA "1 1 1"
      fogType         (LINEAR|EXPONENTIAL) "LINEAR"
      visibilityRange CDATA "0"
      bind            (true|false) "false"
      bindTime        CDATA "-1"
      isBound         (true|false) "false"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT FontStyle EMPTY >
<!ATTLIST FontStyle
      family      CDATA "SERIF"
      horizontal  (true|false) "true"
      justify     CDATA "BEGIN"
      language    CDATA #IMPLIED
```

```
      leftToRight (true|false) "true"
      size CDATA  "1.0"
      spacing     CDATA "1.0"
      style       (PLAIN|BOLD|ITALIC) "BOLD"
      topToBottom (true|false) "true"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Group %Children; >
<!ATTLIST Group
      bboxCenter CDATA "0 0 0"
      bboxSize   CDATA "-1 -1 -1"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT ImageTexture EMPTY >
<!ATTLIST ImageTexture
      url     CDATA #IMPLIED
      repeatS (true|false) "true"
      repeatT (true|false) "true"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT IndexedFaceSet ( ( Color?, Coordinate?, Normal?,
TextureCoordinate? ) |
                           ( Coordinate?, Color?, Normal?,
TextureCoordinate? ) ) >
<!ATTLIST IndexedFaceSet
      ccw             (true|false) "true"
      colorIndex      CDATA #IMPLIED
      colorPerVertex  (true|false) "true"
      convex          (true|false) "true"
      coordIndex      CDATA #IMPLIED
      creaseAngle     CDATA "0"
      normalIndex     CDATA #IMPLIED
      normalPerVertex (true|false) "true"
      solid           (true|false) "true"
      texCoordIndex   CDATA #IMPLIED
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT IndexedLineSet ( ( Color?, Coordinate? ) |
                           ( Coordinate?, Color? ) ) >
<!ATTLIST IndexedLineSet
      colorIndex      CDATA #IMPLIED
      colorPerVertex  (true|false) "true"
      coordIndex      CDATA #IMPLIED
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Inline ( %SceneNodes; ) >
<!ATTLIST Inline
      url         CDATA #REQUIRED
      bboxCenter  CDATA "0 0 0"
      bboxSize    CDATA "-1 -1 -1"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT LOD %Children; >
<!ATTLIST LOD
      center CDATA "0 0 0"
      range  CDATA #IMPLIED
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>
```

```
<!ELEMENT Material EMPTY >
<!ATTLIST Material
      ambientIntensity CDATA "0.2"
      diffuseColor     CDATA "0.8 0.8 0.8"
      emissiveColor    CDATA "0 0 0"
      shininess        CDATA "0.2"
      specularColor    CDATA "0 0 0"
      transparency     CDATA "0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT MovieTexture EMPTY >
<!ATTLIST MovieTexture
      loop      (true|false) "true"
      speed     CDATA "1.0"
      startTime CDATA "0"
      stopTime  CDATA "0"
      url       CDATA #IMPLIED
      repeatS   (true|false) "true"
      repeatT   (true|false) "true"
      duration  CDATA "0.0"
      isActive  (true|false) "false"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT NavigationInfo EMPTY >
<!ATTLIST NavigationInfo
      avatarSize      CDATA "0.25 1.6 0.75"
      headlight       (true|false) "true"
      speed           CDATA "1"
      type            NMTOKEN #IMPLIED
      visibilityLimit CDATA "0"
      bind            (true|false) "false"
      bindTime        CDATA "-1"
      isBound         (true|false) "false"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Normal EMPTY >
<!ATTLIST Normal
      vector    CDATA #REQUIRED
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT NormalInterpolator EMPTY >
<!ATTLIST NormalInterpolator
      key      CDATA #IMPLIED
      keyValue CDATA #IMPLIED
      fraction CDATA "0"
      value    CDATA "0 0 0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT OrientationInterpolator EMPTY >
<!ATTLIST OrientationInterpolator
      key      CDATA #IMPLIED
      keyValue CDATA #IMPLIED
      fraction CDATA "0"
      value    CDATA "0 0 1 0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT PixelTexture EMPTY >
```

```
<!ATTLIST PixelTexture
      image   CDATA "0 0 0"
      repeatS (true|false) "true"
      repeatT (true|false) "true"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT PlaneSensor EMPTY >
<!ATTLIST PlaneSensor
      autoOffset  (true|false) "true"
      enabled     (true|false) "true"
      maxPosition CDATA "-1 -1"
      minPosition CDATA "0 0"
      offset      CDATA "0 0 0"
      isActive    (true|false) "false"
      trackPoint  CDATA "0 0 0"
      translation CDATA "0 0 0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT PointLight EMPTY >
<!ATTLIST PointLight
      ambientIntensity CDATA "0"
      attenuation      CDATA "1 0 0"
      color            CDATA "1 1 1"
      intensity        CDATA "1"
      location         CDATA "0 0 0"
      on               (true|false) "true"
      radius           CDATA "100"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT PointSet ( ( (Color?), (Coordinate?) ) |
                     ( (Coordinate?), (Color?) ) ) >
<!ATTLIST PointSet
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT PositionInterpolator EMPTY >
<!ATTLIST PositionInterpolator
      key      CDATA #IMPLIED
      keyValue CDATA #IMPLIED
      fraction CDATA "0"
      value    CDATA "0 0 0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT ProximitySensor EMPTY >
<!ATTLIST ProximitySensor
      center      CDATA "0 0 0"
      size        CDATA "0 0 0"
      enabled     (true|false) "true"
      isActive    (true|false) "false"
      position    CDATA "0 0 0"
      orientation CDATA "0 0 1 0"
      enterTime   CDATA "0"
      exitTime    CDATA "0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Proxy ( %CollisionProxyNodes; )? >
<!ATTLIST Proxy
      DEF ID      #IMPLIED
      USE IDREF   #IMPLIED >
```

```
<!ELEMENT ScalarInterpolator EMPTY >
<!ATTLIST ScalarInterpolator
      key      CDATA #IMPLIED
      keyValue CDATA #IMPLIED
      fraction CDATA "0"
      value    CDATA "0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>


<!-- [rg] I still have the hardest time calling compiled bytecode Script
-->
<!-- plus the content model should be something like (Field+ , #PCDATA)
but -->
<!-- that is illegal. We need to resolve these.-->

<!ELEMENT Script (Field+) >
<!ATTLIST Script
      url          CDATA #IMPLIED
      directOutput (true|false) "false"
      mustEvaluate (true|false) "false"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Shape ( ( Appearance?, (%GeometryNodes;)? ) |
                  ( (%GeometryNodes;)?, Appearance? ))   >
<!ATTLIST Shape
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Sound ( AudioClip | MovieTexture ) >
<!ATTLIST Sound
      direction  CDATA "0 0 1"
      intensity  CDATA "1"
      location   CDATA "0 0 0"
      maxBack    CDATA "10"
      maxFront   CDATA "10"
      minBack    CDATA "1"
      minFront   CDATA "1"
      priority   CDATA "0"
      spatialize (true|false) "true"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Sphere EMPTY >
<!ATTLIST Sphere
      radius CDATA "1"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT SphereSensor EMPTY >
<!ATTLIST SphereSensor
      autoOffset (true|false) "true"
      enabled    (true|false) "true"
      offset     CDATA "0 1 0 0"
      isActive   (true|false) "false"
      rotation   CDATA "0 1 0 0"
      trackPoint CDATA "0 0 0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT SpotLight EMPTY >
<!ATTLIST SpotLight
```

```
      ambientIntensity CDATA "0"
      attenuation      CDATA "1 0 0"
      beamWidth        CDATA "1.570796"
      color            CDATA "1 1 1"
      cutOffAngle      CDATA ".785398"
      direction        CDATA "0 0 -1"
      intensity        CDATA "1"
      location         CDATA "0 0 0"
      on               (true|false) "true"
      radius           CDATA "100"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Switch  %Children; >
<!ATTLIST Switch
      whichChoice CDATA "-1"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Text ( FontStyle? ) >
<!ATTLIST Text
      string    CDATA #IMPLIED
      length    CDATA #IMPLIED
      maxExtent CDATA "0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT TextureCoordinate EMPTY >
<!ATTLIST TextureCoordinate
      point  CDATA    #IMPLIED
      DEF    ID       #IMPLIED
      USE    IDREF    #IMPLIED>

<!ELEMENT TextureTransform EMPTY >
<!ATTLIST TextureTransform
      center      CDATA "0 0"
      rotation    CDATA "0"
      scale       CDATA "1 1"
      translation CDATA "0 0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT TimeSensor EMPTY >
<!ATTLIST TimeSensor
      cycleInterval CDATA "1.0"
      enabled       (true|false) "true"
      loop          (true|false) "false"
      startTime     CDATA "0"
      stopTime      CDATA "0"
      cycleTime     CDATA "0"
      fraction      CDATA "0"
      isActive      (true|false) "false"
      time          CDATA "0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT TouchSensor EMPTY >
<!ATTLIST TouchSensor
      enabled      (true|false) "true"
      hitNormal    CDATA "0 0 1"
      hitPoint     CDATA "0 0 0"
      hitTexCoord  CDATA "0 0"
      isActive     (true|false) "false"
      isOver       (true|false) "false"
```

```
      touchTime    CDATA "0"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Transform %Children; >
<!ATTLIST Transform
      center           CDATA "0 0 0"
      rotation         CDATA "0 0 1 0"
      scale            CDATA "1 1 1"
      scaleOrientation CDATA "0 0 1 0"
      translation      CDATA "0 0 0"
      bboxCenter       CDATA "0 0 0"
      bboxSize         CDATA "-1 -1 -1"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED >

<!ELEMENT Viewpoint EMPTY >
<!ATTLIST Viewpoint
      fieldOfView CDATA "0.785398"
      jump        (true|false) "true"
      orientation CDATA "0 0 1 0"
      position    CDATA "0 0 10"
      description CDATA #IMPLIED
      bind        (true|false) "false"
      bindTime    CDATA "-1"
      isBound     (true|false) "false"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT VisibilitySensor EMPTY >
<!ATTLIST VisibilitySensor
      center    CDATA "0 0 0"
      enabled   (true|false) "true"
      size      CDATA "0 0 0"
      enterTime CDATA "0.0"
      exitTime  CDATA "0.0"
      isActive  (true|false) "false"
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT WorldInfo EMPTY >
<!ATTLIST WorldInfo
      info  CDATA #IMPLIED
      title CDATA #IMPLIED
      DEF ID       #IMPLIED
      USE IDREF    #IMPLIED>

<!ELEMENT Field EMPTY >
<!ATTLIST Field
      name  ID       #REQUIRED
      value CDATA        #IMPLIED
      type  NMTOKEN      #IMPLIED
      IS    NMTOKEN        #IMPLIED>

<!-- from and to fields need to be nmtokes incase the field is using its
IS alias. no
      multi ID's allowed for elements -->
<!ELEMENT Route EMPTY >
<!ATTLIST Route
      fromNode    IDREF #REQUIRED
      toNode      IDREF #REQUIRED
      fromField   NMTOKEN     #REQUIRED
      toField     NMTOKEN     #REQUIRED>
```

```
<!-- if url is present, ChildrenNodes are ignored -->
<!ELEMENT ProtoDefine ((Field)*, (%ChildrenNodes;)*) >
<!ATTLIST ProtoDefine
      type NMTOKEN    #REQUIRED
      url  CDATA  #IMPLIED>


<!ELEMENT Proto ( Field )*>
<!ATTLIST Proto
      DEF  ID            #IMPLIED
      USE  IDREF     #IMPLIED
      type NMTOKEN   #REQUIRED>


<!-- Example PROTO Usage, external proto:
# some file a.xml with definition
# <ProtoDefine type="myBox">
#    <Field name="yourSize" type="SFVec3f" value="1 1 2">
#    <Shape>
#        <Box size="yourSize">
#    </Shape>
# </ProtoDefine>
#
# then somewhere else in a document b.xml with new default value for
mySize
# <ProtoDefine type="myBox" url="a.xml">
#    <Field name="mySize" type="SFVec3f" IS="yourSize" value="2 1 2"/>
# </ProtoDefine>
#
# then the first instance in b.xml
# if the IS field is not declared, then the name is used directly
# <Group>
#   <Proto type="myBox" DEF="aTwoThreeTenBox" >
#     <Field name="mySize" value="2 3 10"/>
#   </Proto>
# </Group>
#
# and thereafter
#
# <Transform translation="1 0 0">
#     <Proto USE="aTwoThreeTenBox"/>
# </Transform>
#
-->
<!-- pretty html generated with code2html.pl 0.6.2 by
ppalfrad@cosy.sbg.ac.at -->
```