EURECOM

3D GRAPH COURSE

LAB N°2 : VRML

Author: **P. Gros,.**
Environment: **VRML, X3Dedit**
.

Introduction

The goal of this TP is to experiment the concepts presented in the course. During this TP you are going to use an interactive 3D rendering program that will allow you to describe 3D scenes using an XML tool that generates VRML scenes. You will experiment how to interact and navigate interactively in the scene.

VRML (Virtual reality modeling language) is a normalized ASCII file format that allows you to describe 3D words on the web. The current version of VRML is 2.0 and you can install a plugin on your favorite web browser in order to automatically show in 3D the content of the file. VRML is based on the B-Rep modeling and generally, the VRML viewers are based on the Z-Buffer algorithm.

During this TP you are going to use 2 softwares :

- **X3Dedit** is a 3D scene editor that creates scene in the X3D format. X3Dedit provides a graphical interface that helps you to define the scene hierarchy. It has an internal fileformat called X3D, X3D format is a XMLized version of VRML. In that format a scene is described as a hierarchy of nodes. X3Dedit includes an option that allows you to automatically generates the VRML using a file transformation based on XSLT. You can access to X3Dedit by running

  ```
  (AllPrograms / X3D Extensible 3D Graphique /X3D-Edit / X3D-Edit)
  ```

- **Cortona** is an internet explorer plugin that handles the visualization of 3D virtual worlds described using the VRML file format. In order to view a 3D virtual world you only have to open a VRML file (which should have the extension ".wrl". The VRML file may be created using any text editor

(emacs, vi, wordpad etc…). To help you in creating this file you may use tools that automatically generates the VRML content from an graphical interface.

In order to use X3Dedit you have to run it and then to open a X3D file `(file/ Open)` and to visualize the scene by transforming it using XSL `(tools / process XSL)` The software configuration automatically calls Internet explorer and launches cortona. You can then see the 3D scene from a default viewpoint, and you may move yourself in the virtual world using the mouse. **BE AWARE that you should open the files using a regular U: path not the UNC \\homes.eurecom.fr.**

You may find the complete VRML documentation at the following address :

https://web.archive.org/web/20140210074407/http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/

http://accad.osu.edu/~pgerstma/class/vnv/resources/info/AnnotatedVrmlRef/Book.html.

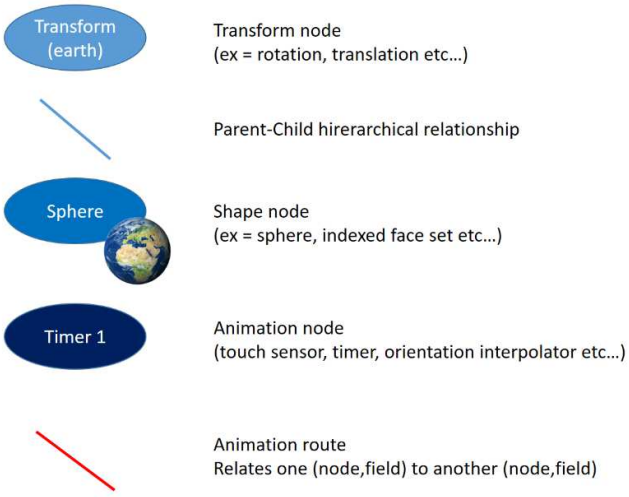
The X3Dedit software is selfdocumented using tooltips. You may find additional inform ation about X3D here :

http://www.web3d.org/

The files described in that document are accessible at the following location : `\\datas\teaching\courses\3DGraph\Lab2_vrml`.

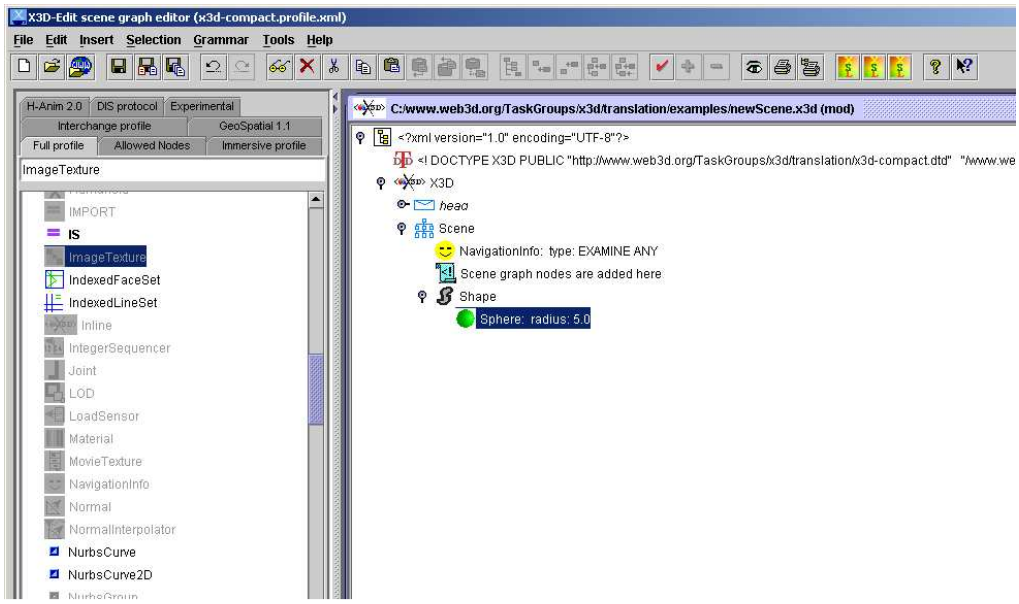
Subject

You will send your answers to each question by email to gros@eurecom.fr with the subject message message being “VRML TP”. You will attach the “.x3d” files of each steps to your message. The TP is structured in one big exercise split into several steps. The objective is to define a solar system and navigate in it. Most of the time, the questions will make use of figure to show the structure of the hierarchy to be obtained. The following symbols are used in the figures:



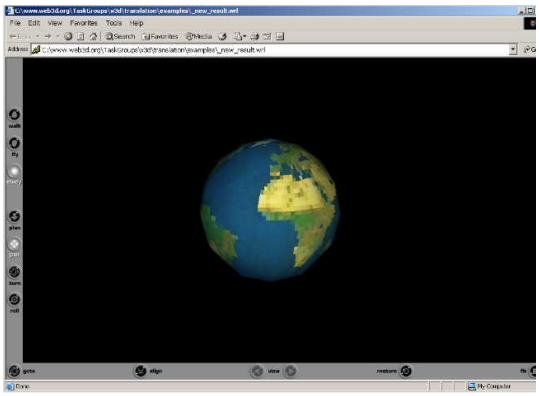
Step N°1 :

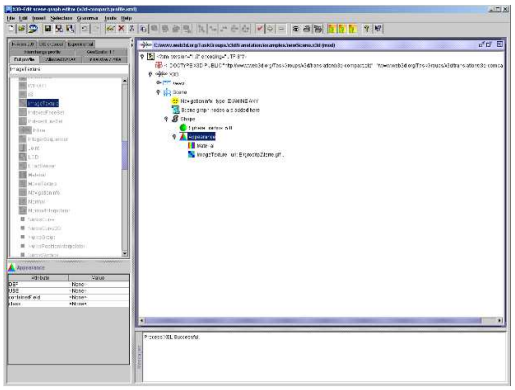
Using X3Dedit please create a "`step1.x3d`" file that includes a sphere of radius 5 at the center of the coordinate system (OC = WC). You can take the `TPvrml.x3d` file as an example. At each step of the TP you will have to create a x3d file called "stepX.x3d" where X is the step number.



Step N°2 :

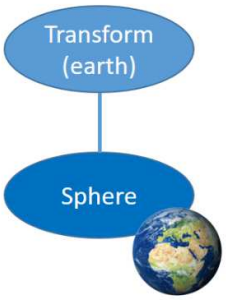
Please Map a texture image called "`terre.gif`" on the previous sphere. Again you may take the TPvrml.x3d file as an example.





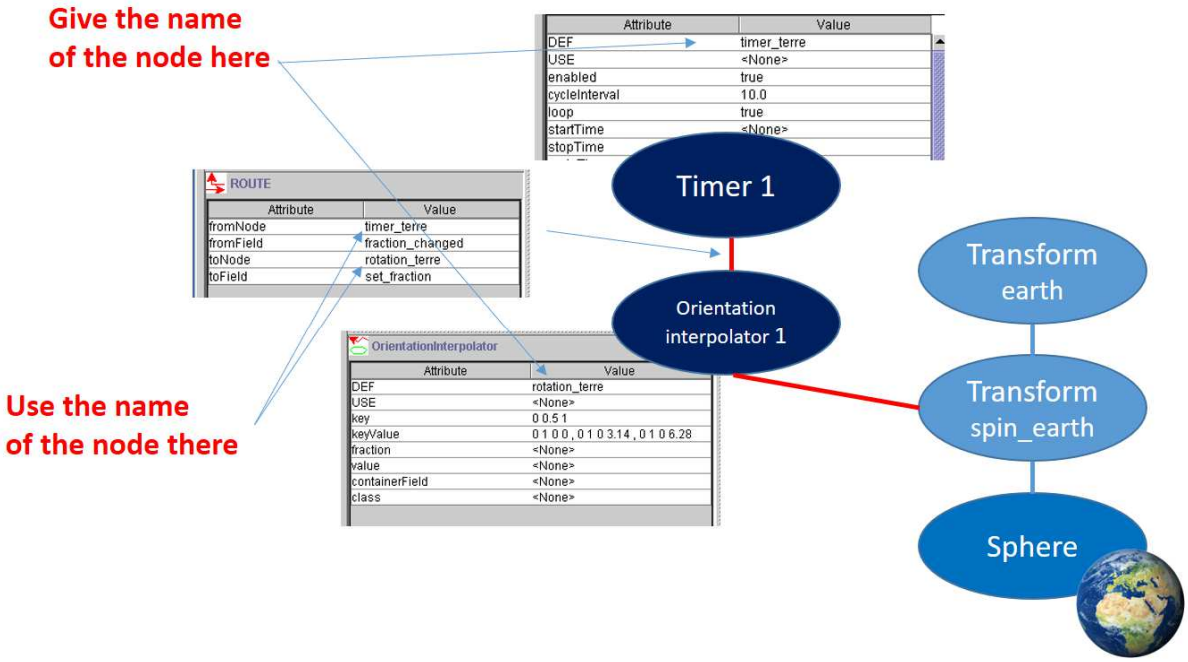
Step N°3 :

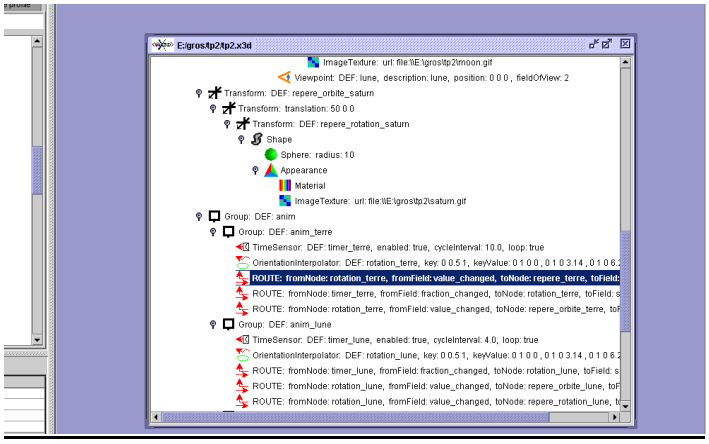
Please add a local coordinate system on top of the sphere object. For this purpose you will have to add a “transform” node (called “earth” in the exemple bellow) and set the sphere as on of its children. The hierarchy will look like this:



Step N°4 :

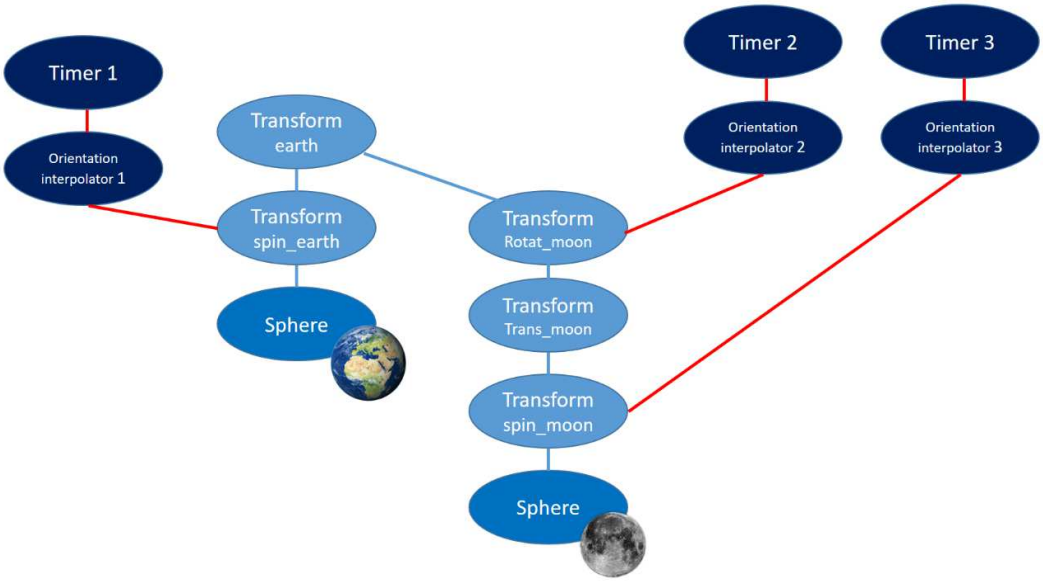
Insert a new transform between the “earth” and the sphere (called “spin_earth” in the exemple bellow). The idea is to have the earth coordinate system (“earth”) independent of the self rotation of the earth so that the moon is not affected by this self rotation. You have now to animate the sphere that is to give it a dynamic behaviour. The sphere should turn around itself at the center of the coordinate system. For that purpose you will have to create a “timesensor” object, a “OrientationInterpolator” and two “routes”. The first route relates the output of the timer to the input of the orientation interpolator, the second relates the output of the orientation interpolator to the input of a new transform node that you have to add on top of the previous one (the one you added in step3).





Step N°5 :

You have to add a new sphere (radius = 2) that gravitate around the previous one and turn around itself as the moon. For that purpose, you have to add a new textured sphere (texture is “moon.gif”). Put the moon in a local coordinate system (the spin_moon for the spin rotation) and put another coordinate system on top of it in order to translate the moon of 7 from the previous one (trans_moon) and a last (rotat_moon) in order to produce the orbital rotation around the earth. The moon sub-hierarchy should be attached to the earth system (earth) and not the global one so that the moon inherits of any movement of the earth. You then have to add 2 new time sensors and 2 new orientation interpolators in order to animate the moon around the earth using appropriate routes. You should obtain the following configuration:



Step N°6 :

Inverse the order in which the “orbit_translation” and the “orbit_rotation” are applied to the moon. What happens when you do this inversion ? explain why.

Step N°7 :

You are then going to use a new kind of geometric object (an indexed faceset) in order to create a simplified version of the space shuttle. You have to create an indexfaceset object and define its geometry using a already defined shape. For that purpose we have already downloaded VRML’s objects on the web (“shuttle.wrl”). You have to edit that file using a text editor such as wordpad.exe and copy/paste the geometry information (ie the point array and the coordinateindex array) in the x3d scene. For that purpose you have to :

1. Copy the content of the coordIndex Array (without the “[ ]”) to the coord Index field of the IndexFaceSet node.

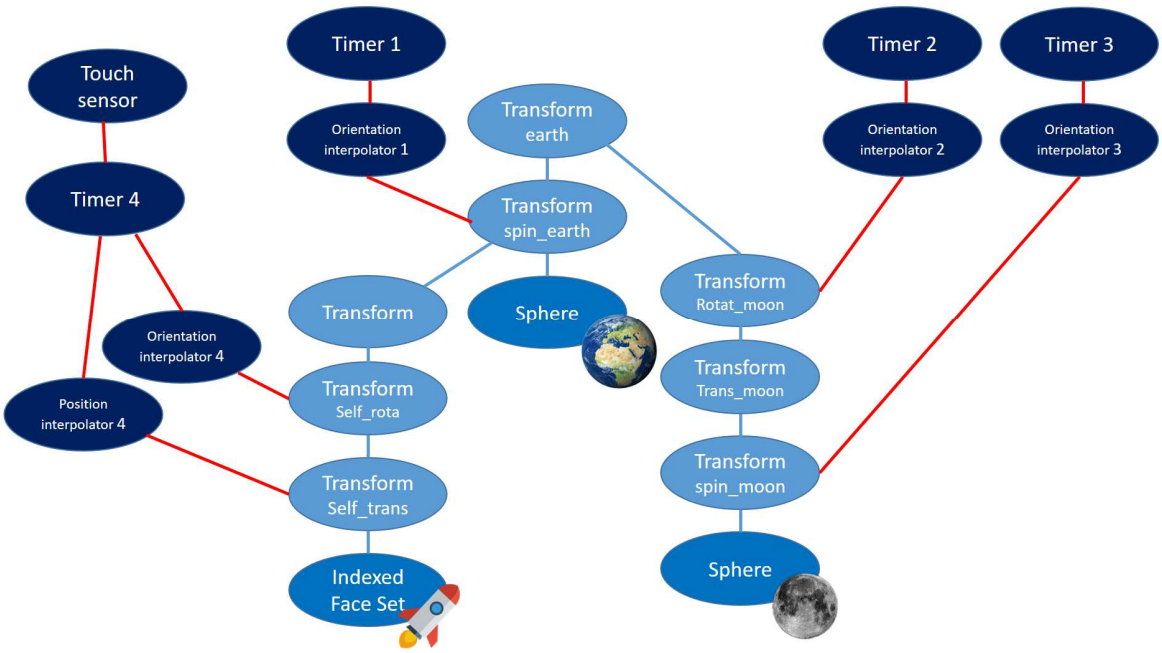
```
10          }
11          IndexedFaceSet {
12           coordIndex  [ 0, 1, 2, -1,
13                         3, 1, 0, -1,
14                         4, 1, 3, -1,
15                         5, 1, 4, -1,
16                         6, 1, 5, -1,
ensible Markup Language file    length : 21236   lines : 705
```

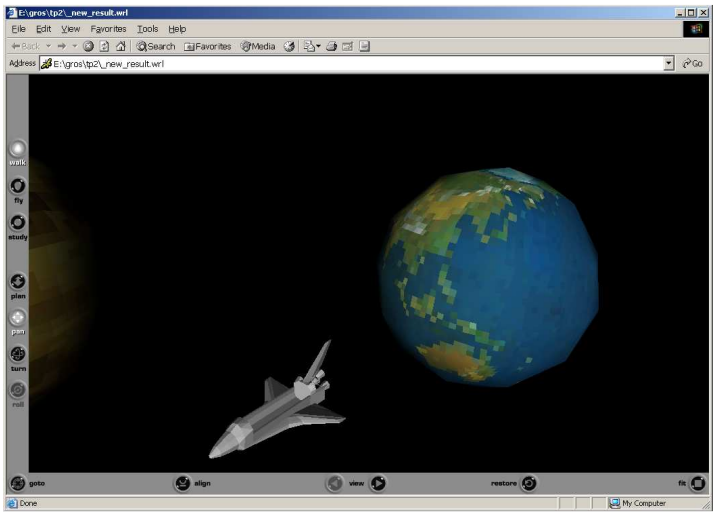
2. Create a Coordinate Node as a child of the IndexFaceSet node, and copy the values of the Point array (without the “[ ]”) to the point field of this new Coordinate Node.

```
10      }
11      Separator {
12          renderCulling   OFF
13          Coordinate3 {
14          point   [ -72.612000 -2.130000 1.024000,
15                    -73.217003 -3.414000 0.000000,
16                    -72.655998 -1.791000 0.000000,
17                    -72.488998 -3.060000 1.536000,
18                    -72.355003 -4.076000 1.365000,
19                    -72.265999 -4.752000 0.512000,
20                    -72.265999 -4.752000 -0.513000,
```

Once you created the geometry you have to give it a reasonable size and put it into the scene, the back of the shuttle being at the earth surface its front pointed to the space.

The objective is to have the shuttle takeoff for a short trip in the space and get back at its original position each time the user clicks on the earth planet. You have thus to put the shuttle in a local coordinate system and attach that local coordinate system to the spin transform of the earth so that the shuttle stays at its position. You also have to introduce at least two transform in order to drive the shuttle during its trip in space. You then have to create the behaviour objects (one touchsensor, one timesensor, one orientation interpolator and one position interpolator). The touchsensor will be put near to the earth node so that a click on the earth activates it. Its output has to be routed to the timesensor, the output of the timesensor being routed to the input of the orientation and translation interpolators.





Step N°8 :

Click on the earth planet, if everything is ok, the shuttle takes off. Please explain its trajectory? You then have to configure the scene so that as soon as it takes off, the shuttle has a linear trajectory in the space.

Step N°9 :

One of the most important aspect of interactive 3D user interfaces is navigation. Navigation is the set of methods provided by the system to allow the user to move into the virtual world. Cortona provides some navigation metaphors that the user may control using its mouse. Please experiment and list the navigation tools supported by cortona, and give a short report (characteristics/advantages/defaults) about their usability.

Step N°10 :

VRML (and thus X3D) supports the concept of viewpoints. Viewpoints are predefined user position that you may jump to using the user interface (you should have pointed out that capability in the previous step). You may attach viewpoints to objects in the scene and thus see the virtual world from different perspectives. Please attach viewpoints to the earth and the moon and the global coordinate system, modify some of their parameters and jump to them. What do you notice ? Is it usable ? what do you think should be done in order for viewpoints to be usable for the user.

Step N°11 :

Now that you have experimented how X3D and VRML works, you have to build a simplified version of the solar system. Please do the following:

- The sun turns around it self at the center of the world.
- Add Mercury, Earth, Mars, Jupiter and Saturn that turn around the Sun.
- The moon turns around the earth.

Step N°12 : (optional, for ex æquo ☺)

You may find other starship models in the TP directory, use them in order to create a group of space vessels in orbit around Jupiter and imagine a behaviour for the group when you click on the sun.

ANNEX (VRML format)

1-Introduction

This annex gives you some highlights about the VRML format (and thus X3d) of the objects you will have to use during this TP. You may find a complete VRML repository at the http://www.web3d.org/vrml/vrml.htm location. You may find the VRML documentation at http://www.web3d.org/Specifications/, a good tutorial is available at http ://www-cern.ch/vrmltut. This annex describes the VRML format, The fields that you may find in the X3d nodes are the same that the one you may find in VRML.

Every VRML file should begins with (no applicable in X3D):

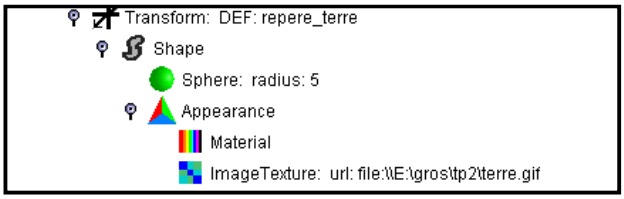
```
#VRML V2.0 utf8
```

2 - Nodes, fields et event

Nearly all the VRML objects (light sources, camera, textures, objects) are nodes. Each node is made up of a set of fields, each field as a name, a type and a value (each field as a default value). Each node also has an interface that is able to send and receive events. This interface is described in terms of eventin and eventout. A special kind of fields is called the exposedfield. An exposed field is a field and on eventin and one eventout: for example, the exposedfield “toto” is a field “toto” plus an event in called “set_toto” and a eventout called “toto_changed”.

Some fields of a node may be themselves nodes (the type of the field is SFNode), you only have to insert a child node with the good type in the father node. For exemple the shape node has a fields which is a geometry node, the VRML syhntax for that is :

```
Shape {
      geometry Sphere { }
}
```

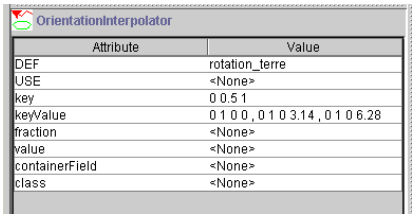


It is possible to give a name to any VRML node. This name has two purposes:

1. You may reuse that object by simply instanciate a other time that object (without describing it again)
2. You SHOULD name an object that you want to connect to other objects using routes (the usual way to animate objects into the virtual world).

You can name an object using the DEF keyword:

```
DEF MonObjetAMoi Shape {
      geometry Sphere { }
}
```



You may reuse the object using the keyword USE :

```
Shape USE MonObjetAMOI
```

3-Camera

A camera is a node called a viewpoint. It is described by a set of fields, we mainly will use the position and description fields. You may create as many viewpoints as you want.

```
Viewpoint { position 0 0 80
            description "far" }
```

```
Viewpoint {
        eventIn      SFBool     set_bind
        exposedField SFFloat    fieldOfView    0.785398
        exposedField SFBool     jump           TRUE
        exposedField SFRotation orientation    0 0 1 0
        exposedField SFVec3f    position       0 0 10
        field        SFString   description    ""
        eventOut     SFTime     bindTime
        eventOut     SFBool     isBound
}
```

Attribute	Value
DEF	fusee
USE	<None>
description	fusee
position	0 0 0
orientation	<None>
fieldOfView	<None>
jump	<None>
centerOfRotation	<None>
bind	<None>
bindTime	<None>
isBound	<None>
containerField	<None>
class	<None>

4-Objects

Physical objects are nodes of type shapes. A shape has two fields : one is of type appearance (it is used to describe the interactions between the object and the light) the other is of type geometry and is used to describe the geometry of objects.

```
Shape {
      appearance Appearance { }
      geometry Sphere { }
}
```

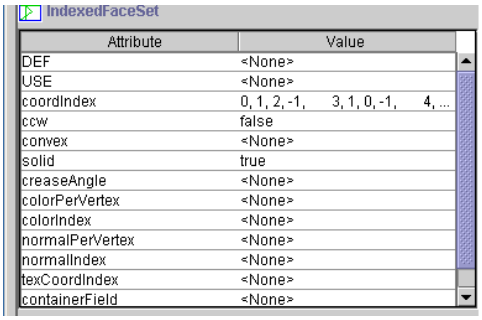
```
Shape {
        exposedField SFNode appearance NULL
        exposedField SFNode geometry   NULL
}
```

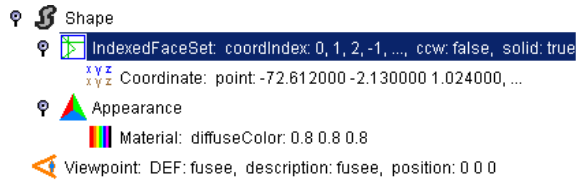
In order to define geometry VRML features a set of primitives (spheres, cones, boxes etc…) and allows the user to describe its own shapes using indexed polygons (Indexedfaceset). An indexfaceset as the following fields :

- The coord field allows the user to define a vertice array.
- The color field allows the user to define a set of colors.
- The coordIndex allows the user to define a set of face based on the vertices. Each face is defined by the indices of its vertices (in the coord array) the end of each face is given by a -1 index. You should create a child node of type coordinate and put the coordinate array here.
- The ccw field is true if the faces are described in the ccw way.
- The solid field is true if the object is volumic (enables backface culling)
- The colorpervertex may be true or false according to the color association strategy.

```
Sphere {
      radius  12.5
}
```

```
IndexedFaceSet {
        color Color { color [ r g b, r g b, …] }
        coord  Coordinate { point [ x y z, x y z, …] }
        coordIndex [ a, b, c, -1, d, e, f, -1 … -1 ]
        field         SFBool  ccw              TRUE
        field         SFBool  colorPerVertex   FALSE
        field         SFBool  convex           TRUE
        field         SFBool  solid            TRUE
}
```





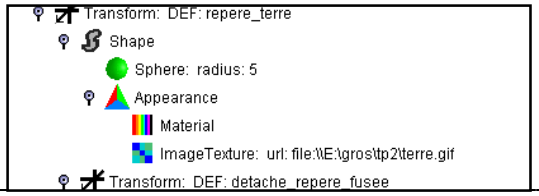
5-Apply a texture to an object

The appearance node is composed of several fields. One of them is of type node and is used to describe the texture to be applied to the object. The child node should be of type Imagetexture.

```
Shape {
      appearance Appearance {
            material Material { }
            texture ImageTexture { url "sun.gif"}
      }
      geometry ...
}
```

```
Appearance {
        exposedField SFNode material          NULL
        exposedField SFNode texture           NULL
        exposedField SFNode textureTransform  NULL
}
```

```
ImageTexture {
        exposedField MFString url     []
        field        SFBool   repeatS TRUE
        field        SFBool   repeatT TRUE
}
```



6-transfom (new local coordinate system)

The transform node allows you to introduce a geometric transformation to its children. It is composed of 3 transformation field (scale, rotation and translation) applied in that order to the children located in the children array. If you wish to do some transforms in another order then the default one you have to chain several transforms.

```
Transform {
      translation 0 0 7
      rotation 0 0 1 3.14
      scale 1 1 1
      children [
            Shape { ...
            }
      ]
}
```

```
Transform {
         eventIn      MFNode      addChildren
         eventIn      MFNode      removeChildren
         exposedField SFVec3f     center           0 0 0
         exposedField MFNode      children         []
         exposedField SFRotation  rotation         0 0 1 0
         exposedField SFVec3f     scale            1 1 1
         exposedField SFRotation  scaleOrientation 0 0 1 0
         exposedField SFVec3f     translation      0 0 0
         field        SFVec3f     bboxCenter       0 0 0
         field        SFVec3f     bboxSize         -1 -1 -1
}
```

7-Describe a Behaviour

In order to describe the dynamic behaviour of an object some fields of that object should be modify along with time. VRML proposes a mechanism based on events. The idea is to have a chain of nodes linked using routes. Along the chain each node receives events from the previous node and sends events to the next node. The typical chain is made of a timer, am interpolator and a target object (or transform):

- The timer (a timesensor) has a cycle time and it regularly generates an eventout that says the proportion of time spent from the previous eventout (as a fraction of the timer cycle time). When the cycle time is elapsed the timer either stops or restart from the beginning.
- The interpolator receives the eventout of the timesensor and computes an output value that is obtained as an interpolation between key values. The key values are given at the node creation as two arrays. There are interpolators for several type of data such orientation, position, color…
- The object (or one of its transform upper in the hierarchy) receives an event and modifies one of its field accordingly. The next images are computed using that value.

8- TimerSensor, OrientationInterpolator and ROUTE

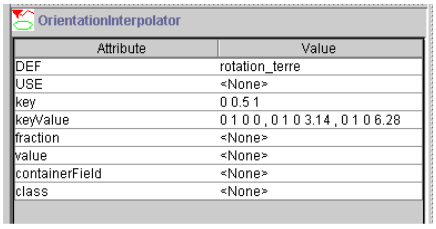
The timeSensor node has a field (cycleInterval) that allow the user to specify the cycle time of the timer in seconds. The loop field syas if the timer should stops or restart from the beginning once it has reached the end. The timer will output events through its fraction_changed eventout. The value are inside the 0,1 interval (0 means starting time, 1 means end of the cycle time, 0,5 means half of the cycle time…). Because timers

are used in routes they should have a name. The following example defines a eternal timer with a period of 4 seconds:

```
DEF MonTimer TimeSensor {
      cycleInterval 4
      loop TRUE
}

TimeSensor {
        exposedField SFTime   cycleInterval 1
        exposedField SFBool   enabled       TRUE
        exposedField SFBool   loop          FALSE
        exposedField SFTime   startTime     0
        exposedField SFTime   stopTime      0
        eventOut     SFTime   cycleTime
        eventOut     SFFloat  fraction_changed
        eventOut     SFBool   isActive
        eventOut     SFTime   time
}
```

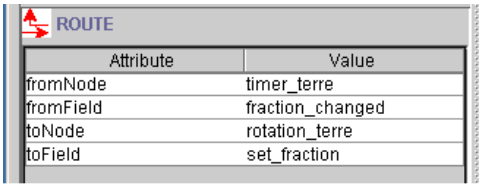
Here is an example of an orientation interpolator used to modify the rotation field of a transform node. The interpolator node has a name in order to used in a route. The key field is an array of key values between 0 and 1. The keyvalue is an array of rotation (an axis and an angle). The size of the key and keyvalues arrays should be the same since each key should correspond to each keyvalue, the interpolator computes the intermediate values using linear interpolation. The event out of the timer is fraction_changed, the event in of the interpolator is set_fraction, the event out is value_changed.




```
DEF MonOrientationInterpolator OrientationInterpolator {
        key [ 0 0.5 1 ]
        keyValue [ 0 1 0 0, 0 1 0 3.14, 0 1 0 6.28 ]
 }

OrientationInterpolator {
        eventIn      SFFloat    set_fraction
        exposedField MFFloat    key           []
        exposedField MFRotation keyValue      []
        eventOut     SFRotation value_changed
}
```

In order to connect the timer to the interpolator you only have to define a route that connects the fraction_changed of the timer to the set_fraction of the interpolator.




```
ROUTE MonTimer.fraction_changed TO MonOrientationInterpolator.set_fraction
```

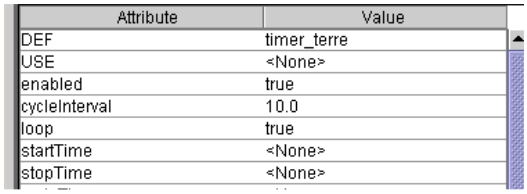
In order to connect the event out of the interpolator (value_changed) to the rotation event in of a transform you should create the following route:

```
ROUTE MonOrientationInterpolator.value_changed TO MaTransformation.set_rotation
```

The process is the same for Position interpolators and translation in the transform node.

9- TouchSensor et TimeSensor

The touchsensor node is used to detect and manage user interactions. When you add a touchsensor as the child of a grouping node (for example a transform) every click of the user on one object that is child of that node is sent back to the touchsensor. The touchsensor may then itself produce an event that we could route to a time sensor in order to start some animation.



Attribute	Value
DEF	timer_terre
USE	<None>
enabled	true
cycleInterval	10.0
loop	true
startTime	<None>
stopTime	<None>

```
Transform {
      children [
            Shape { ...
            }
            DEF MonTouchSensor TouchSensor {}
      ]
}

DEF MonTimeSensor TimeSensor {                          cycleInterval 4
      loop FALSE
}

ROUTE MonTouchSensor.touchTime TO MonTimeSensor.set_startTime
```

```
TouchSensor {
        exposedField SFBool  enabled TRUE
        eventOut     SFVec3f hitNormal_changed
        eventOut     SFVec3f hitPoint_changed
        eventOut     SFVec2f hitTexCoord_changed
        eventOut     SFBool  isActive
        eventOut     SFBool  isOver
        eventOut     SFTime  touchTime
}
```

A QUICK GUIDE TO VRML NODES

```
Anchor {
  eventIn      MFNode   addChildren
  eventIn      MFNode   removeChildren
  exposedField MFNode   children        []
  exposedField SFString description     ""
  exposedField MFString parameter       []
  exposedField MFString url             []
  field        SFVec3f  bboxCenter      0 0 0     # (-,)
  field        SFVec3f  bboxSize        -1 -1 -1  # (0,) or -1,-1,-1
}
Appearance {
  exposedField SFNode material          NULL
  exposedField SFNode texture           NULL
  exposedField SFNode textureTransform  NULL
}
AudioClip {
  exposedField   SFString description      ""
  exposedField   SFBool   loop             FALSE
  exposedField   SFFloat  pitch            1.0        # (0,)
  exposedField   SFTime   startTime        0          # (-,)
  exposedField   SFTime   stopTime         0          # (-,)
  exposedField   MFString url              []
  eventOut       SFTime   duration_changed
  eventOut       SFBool   isActive
}
Background {
  eventIn      SFBool   set_bind
  exposedField MFFloat  groundAngle  []            # [0,/2]
  exposedfield MFColor  groundColor  []            # [0,1]
  exposedField MFString backUrl      []
  exposedField MFString bottomUrl    []
  exposedField MFString frontUrl     []
  exposedField MFString leftUrl      []
  exposedField MFString rightUrl     []
  exposedField MFString topUrl       []
  exposedField MFFloat  skyAngle     []            # [0,]
  exposedField MFColor  skyColor     [ 0 0 0 ]     # [0,1]
  eventOut     SFBool   isBound
}
Billboard {
  eventIn      MFNode   addChildren
  eventIn      MFNode   removeChildren
  exposedField SFVec3f  axisOfRotation  0 1 0      # (-,)
  exposedField MFNode   children        []
  field        SFVec3f  bboxCenter      0 0 0      # (-,)
  field        SFVec3f  bboxSize        -1 -1 -1   # (0,) or -1,-1,-1
}
Box {
  field    SFVec3f size  2 2 2        # (0, )
}



Collision {
  eventIn      MFNode   addChildren
  eventIn      MFNode   removeChildren
  exposedField MFNode   children        []
  exposedField SFBool   collide         TRUE
  field        SFVec3f  bboxCenter      0 0 0      # (-,)
  field        SFVec3f  bboxSize        -1 -1 -1   # (0,) or -1,-1,-1
  field        SFNode   proxy           NULL
  eventOut     SFTime   collideTime
}
Color {
  exposedField MFColor color  []         # [0,1]
}


ColorInterpolator {
  eventIn      SFFloat set_fraction        # (-,)
  exposedField MFFloat key           []    # (-,)
  exposedField MFColor keyValue      []    # [0,1]
  eventOut     SFColor value_changed
}
Cone {
  field     SFFloat   bottomRadius 1        # (0,)
  field     SFFloat   height       2        # (0,)
  field     SFBool    side         TRUE
  field     SFBool    bottom       TRUE
}
Coordinate {
  exposedField MFVec3f point  []      # (-,)
}
CoordinateInterpolator {
  eventIn      SFFloat set_fraction        # (-,)
  exposedField MFFloat key           []    # (-,)
  exposedField MFVec3f keyValue      []    # (-,)
  eventOut     MFVec3f value_changed
}
Cylinder {
  field    SFBool    bottom  TRUE
  field    SFFloat   height  2         # (0,)
  field    SFFloat   radius  1         # (0,)
```

```
  field    SFBool    side    TRUE
  field    SFBool    top     TRUE
}
CylinderSensor {
  exposedField SFBool     autoOffset TRUE
  exposedField SFFloat    diskAngle  0.262       # (0,/2)
  exposedField SFBool     enabled    TRUE
  exposedField SFFloat    maxAngle   -1          # [-2,2]
  exposedField SFFloat    minAngle   0           # [-2,2]
  exposedField SFFloat    offset     0           # (-,)
  eventOut     SFBool     isActive
  eventOut     SFRotation rotation_changed
  eventOut     SFVec3f    trackPoint_changed
}
DirectionalLight {
  exposedField SFFloat ambientIntensity  0        # [0,1]
  exposedField SFColor color             1 1 1    # [0,1]
  exposedField SFVec3f direction         0 0 -1   # (-,)
  exposedField SFFloat intensity         1        # [0,1]
  exposedField SFBool  on                TRUE
}
ElevationGrid {
  eventIn      MFFloat  set_height
  exposedField SFNode   color             NULL
  exposedField SFNode   normal            NULL
  exposedField SFNode   texCoord          NULL
  field        MFFloat  height            []      # (-,)
  field        SFBool   ccw               TRUE
  field        SFBool   colorPerVertex    TRUE
  field        SFFloat  creaseAngle       0       # [0,]
  field        SFBool   normalPerVertex   TRUE
  field        SFBool   solid             TRUE
  field        SFInt32  xDimension        0       # [0,)
  field        SFFloat  xSpacing          1.0     # (0,)
  field        SFInt32  zDimension        0       # [0,)
  field        SFFloat  zSpacing          1.0     # (0,)
}
Extrusion {
  eventIn MFVec2f    set_crossSection
  eventIn MFRotation set_orientation
  eventIn MFVec2f    set_scale
  eventIn MFVec3f    set_spine
  field   SFBool     beginCap         TRUE
  field   SFBool     ccw              TRUE
  field   SFBool     convex           TRUE
  field   SFFloat    creaseAngle      0                # [0,)
  field   MFVec2f    crossSection     [ 1 1, 1 -1, -1 -1,
                                       -1 1, 1  1 ]    # (-,)
  field   SFBool     endCap           TRUE
  field   MFRotation orientation      0 0 1 0          # [-1,1],(-,)
  field   MFVec2f    scale            1 1              # (0,)
  field   SFBool     solid            TRUE
  field   MFVec3f    spine            [ 0 0 0, 0 1 0 ] # (-,)
}
Fog {
  exposedField SFColor  color           1 1 1      # [0,1]
  exposedField SFString fogType         "LINEAR"
  exposedField SFFloat  visibilityRange 0          # [0,)
  eventIn      SFBool   set_bind
  eventOut     SFBool   isBound
}
FontStyle {
  field MFString family       ["SERIF"]
  field SFBool   horizontal   TRUE
  field MFString justify      "BEGIN"
  field SFString language     ""
  field SFBool   leftToRight  TRUE
  field SFFloat  size         1.0          # (0,)
  field SFFloat  spacing      1.0          # [0,)
  field SFString style        "PLAIN"
  field SFBool   topToBottom  TRUE
}
Group {
  eventIn      MFNode  addChildren
  eventIn      MFNode  removeChildren
  exposedField MFNode  children       []
  field        SFVec3f bboxCenter     0 0 0     # (-,)
  field        SFVec3f bboxSize       -1 -1 -1  # (0,) or -1,-1,-1
}
ImageTexture {
  exposedField MFString url     []
  field        SFBool   repeatS TRUE
  field        SFBool   repeatT TRUE
}
IndexedFaceSet {
  eventIn       MFInt32 set_colorIndex
  eventIn       MFInt32 set_coordIndex
  eventIn       MFInt32 set_normalIndex
  eventIn       MFInt32 set_texCoordIndex
  exposedField  SFNode  color             NULL
  exposedField  SFNode  coord             NULL
  exposedField  SFNode  normal            NULL
  exposedField  SFNode  texCoord          NULL
  field         SFBool  ccw               TRUE
  field         MFInt32 colorIndex        []        # [-1,)
  field         SFBool  colorPerVertex    TRUE
```

```
  field         SFBool  convex           TRUE
  field         MFInt32 coordIndex       []        # [-1,)
  field         SFFloat creaseAngle      0         # [0,)
  field         MFInt32 normalIndex      []        # [-1,)
  field         SFBool  normalPerVertex  TRUE
  field         SFBool  solid            TRUE
  field         MFInt32 texCoordIndex    []        # [-1,)
}
IndexedLineSet {
  eventIn       MFInt32 set_colorIndex
  eventIn       MFInt32 set_coordIndex
  exposedField  SFNode  color            NULL
  exposedField  SFNode  coord            NULL
  field         MFInt32 colorIndex       []     # [-1,)
  field         SFBool  colorPerVertex   TRUE
  field         MFInt32 coordIndex       []     # [-1,)
}
Inline {
  exposedField MFString url        []
  field        SFVec3f  bboxCenter 0 0 0     # (-,)
  field        SFVec3f  bboxSize   -1 -1 -1  # (0,) or -1,-1,-1
}
LOD {
  exposedField MFNode  level    []
  field        SFVec3f center   0 0 0    # (-,)
  field        MFFloat range    []       # (0,)
}

Material {
  exposedField SFFloat ambientIntensity  0.2         # [0,1]
  exposedField SFColor diffuseColor      0.8 0.8 0.8 # [0,1]
  exposedField SFColor emissiveColor     0 0 0       # [0,1]
  exposedField SFFloat shininess         0.2         # [0,1]
  exposedField SFColor specularColor     0 0 0       # [0,1]
  exposedField SFFloat transparency      0           # [0,1]
}
MovieTexture {
  exposedField SFBool   loop             FALSE
  exposedField SFFloat  speed            1.0      # (-,)
  exposedField SFTime   startTime        0        # (-,)
  exposedField SFTime   stopTime         0        # (-,)
  exposedField MFString url              []
  field        SFBool   repeatS          TRUE
  field        SFBool   repeatT          TRUE
  eventOut     SFTime   duration_changed
  eventOut     SFBool   isActive
}
NavigationInfo {
  eventIn      SFBool   set_bind
  exposedField MFFloat  avatarSize      [0.25, 1.6, 0.75] # [0,)
  exposedField SFBool   headlight       TRUE
  exposedField SFFloat  speed           1.0               # [0,)
  exposedField MFString type            ["WALK", "ANY"]
  exposedField SFFloat  visibilityLimit 0.0               # [0,)
  eventOut     SFBool   isBound
}
Normal {
  exposedField MFVec3f vector  []   # (-,)
}
NormalInterpolator {
  eventIn      SFFloat set_fraction       # (-,)
  exposedField MFFloat key           []   # (-,)
  exposedField MFVec3f keyValue      []   # (-,)
  eventOut     MFVec3f value_changed
}
OrientationInterpolator {
  eventIn      SFFloat    set_fraction      # (-,)
  exposedField MFFloat    key           []  # (-,)
  exposedField MFRotation keyValue      []  # [-1,1],(-,)
  eventOut     SFRotation value_changed
}
PixelTexture {
  exposedField SFImage  image      0 0 0    # see "4.5 SFImage"
  field        SFBool   repeatS    TRUE
  field        SFBool   repeatT    TRUE
}
PlaneSensor {
  exposedField SFBool  autoOffset          TRUE
  exposedField SFBool  enabled             TRUE
  exposedField SFVec2f maxPosition         -1 -1     # (-,)
  exposedField SFVec2f minPosition         0 0       # (-,)
  exposedField SFVec3f offset              0 0 0     # (-,)
  eventOut     SFBool  isActive
  eventOut     SFVec3f trackPoint_changed
  eventOut     SFVec3f translation_changed
}

PointLight {
  exposedField SFFloat ambientIntensity  0       # [0,1]
  exposedField SFVec3f attenuation       1 0 0   # [0,)
  exposedField SFColor color             1 1 1   # [0,1]
  exposedField SFFloat intensity         1       # [0,1]
  exposedField SFVec3f location          0 0 0   # (-,)
  exposedField SFBool  on                TRUE
  exposedField SFFloat radius            100     # [0,)
}
```

```
PointSet {
  exposedField  SFNode  color      NULL
  exposedField  SFNode  coord      NULL
}
PositionInterpolator {
  eventIn      SFFloat set_fraction        # (-,)
  exposedField MFFloat key           []    # (-,)
  exposedField MFVec3f keyValue      []    # (-,)
  eventOut     SFVec3f value_changed
}
ProximitySensor {
  exposedField SFVec3f    center      0 0 0    # (-,)
  exposedField SFVec3f    size        0 0 0    # [0,)
  exposedField SFBool     enabled     TRUE
  eventOut     SFBool     isActive
  eventOut     SFVec3f    position_changed
  eventOut     SFRotation orientation_changed
  eventOut     SFTime     enterTime
  eventOut     SFTime     exitTime
}
ScalarInterpolator {
  eventIn      SFFloat set_fraction         # (-,)
  exposedField MFFloat key           []     # (-,)
  exposedField MFFloat keyValue      []     # (-,)
  eventOut     SFFloat value_changed
}
Script {
  exposedField MFString url           []
  field        SFBool   directOutput  FALSE
  field        SFBool   mustEvaluate  FALSE
  # And any number of:
  eventIn      eventType eventName
  field        fieldType fieldName initialValue
  eventOut     eventType eventName
}
Shape {
  exposedField SFNode appearance NULL
  exposedField SFNode geometry   NULL
}



Sound {
  exposedField SFVec3f  direction     0 0 1   # (-,)
  exposedField SFFloat  intensity     1       # [0,1]
  exposedField SFVec3f  location      0 0 0   # (-,)
  exposedField SFFloat  maxBack       10      # [0,)
  exposedField SFFloat  maxFront      10      # [0,)
  exposedField SFFloat  minBack       1       # [0,)
  exposedField SFFloat  minFront      1       # [0,)
  exposedField SFFloat  priority      0       # [0,1]
  exposedField SFNode   source        NULL
  field        SFBool   spatialize    TRUE
}
Sphere {
  field SFFloat radius  1    # (0,)
}
SphereSensor {
  exposedField SFBool     autoOffset        TRUE
  exposedField SFBool     enabled           TRUE
  exposedField SFRotation offset            0 1 0 0  # [-1,1],(-,)
  eventOut     SFBool     isActive
  eventOut     SFRotation rotation_changed
  eventOut     SFVec3f    trackPoint_changed
}
SpotLight {
  exposedField SFFloat ambientIntensity  0         # [0,1]
  exposedField SFVec3f attenuation       1 0 0     # [0,)
  exposedField SFFloat beamWidth         1.570796  # (0,/2]
  exposedField SFColor color             1 1 1     # [0,1]
  exposedField SFFloat cutOffAngle       0.785398  # (0,/2]
  exposedField SFVec3f direction         0 0 -1    # (-,)
  exposedField SFFloat intensity         1         # [0,1]
  exposedField SFVec3f location          0 0 0     # (-,)
  exposedField SFBool  on                TRUE
  exposedField SFFloat radius            100       # [0,)
}
Switch {
  exposedField    MFNode  choice      []
  exposedField    SFInt32 whichChoice -1    # [-1,)
}
Text {
  exposedField  MFString string    []
  exposedField  SFNode   fontStyle NULL
  exposedField  MFFloat  length    []      # [0,)
  exposedField  SFFloat  maxExtent 0.0     # [0,)
}
TextureCoordinate {
  exposedField MFVec2f point  []      # (-,)
}
TextureTransform {
  exposedField SFVec2f center      0 0     # (-,)
  exposedField SFFloat rotation    0       # (-,)
  exposedField SFVec2f scale       1 1     # (-,)
  exposedField SFVec2f translation 0 0     # (-,)
}
```

```
TimeSensor {
  exposedField SFTime   cycleInterval 1       # (0,)
  exposedField SFBool   enabled       TRUE
  exposedField SFBool   loop          FALSE
  exposedField SFTime   startTime     0       # (-,)
  exposedField SFTime   stopTime      0       # (-,)
  eventOut     SFTime   cycleTime
  eventOut     SFFloat  fraction_changed
  eventOut     SFBool   isActive
  eventOut     SFTime   time
}
TouchSensor {
  exposedField SFBool  enabled TRUE
  eventOut     SFVec3f hitNormal_changed
  eventOut     SFVec3f hitPoint_changed
  eventOut     SFVec2f hitTexCoord_changed
  eventOut     SFBool  isActive
  eventOut     SFBool  isOver
  eventOut     SFTime  touchTime
}
Transform {
  eventIn      MFNode      addChildren
  eventIn      MFNode      removeChildren
  exposedField SFVec3f     center           0 0 0    # (-,)
  exposedField MFNode      children         []
  exposedField SFRotation  rotation         0 0 1 0  # [-1,1],(-,)
  exposedField SFVec3f     scale            1 1 1    # (0,)
  exposedField SFRotation  scaleOrientation 0 0 1 0  # [-1,1],(-,)
  exposedField SFVec3f     translation      0 0 0    # (-,)
  field        SFVec3f     bboxCenter       0 0 0    # (-,)
  field        SFVec3f     bboxSize         -1 -1 -1 # (0,) or -1,-1,-1
}

Viewpoint {
  eventIn      SFBool     set_bind
  exposedField SFFloat    fieldOfView    0.785398  # (0,)
  exposedField SFBool     jump           TRUE
  exposedField SFRotation orientation    0 0 1 0   # [-1,1],(-,)
  exposedField SFVec3f    position       0 0 10    # (-,)
  field        SFString   description    ""
  eventOut     SFTime     bindTime
  eventOut     SFBool     isBound
}
VisibilitySensor {
  exposedField SFVec3f center   0 0 0      # (-,)
  exposedField SFBool  enabled  TRUE
  exposedField SFVec3f size     0 0 0      # [0,)
  eventOut     SFTime  enterTime
  eventOut     SFTime  exitTime
  eventOut     SFBool  isActive
}
WorldInfo {
  field MFString info  []
  field SFString title ""
}
```