



VRML: Lights, Events, and Sensors

Day 4: VRML Lesson 3

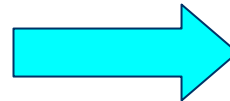
1000 Galliher Dr. | Fairmont, WV 26555-2720

Phone 304.368.9300 | www.wvhtf.org

- Lessons
 - Creating Lights
 - Fields and Routes
 - Events – eventIn and eventOut
 - Touch and Time Sensors
 - Animation (Position Interpolators)
- 2-Day Programming Assignment

- PointLight Node Default Fields
 - Most commonly adjusted fields
 - See Appendix A for full definition

```
PointLight {  
  on      TRUE  
  intensity 1  
  color   1, 1, 1  
  radius  100  
}
```

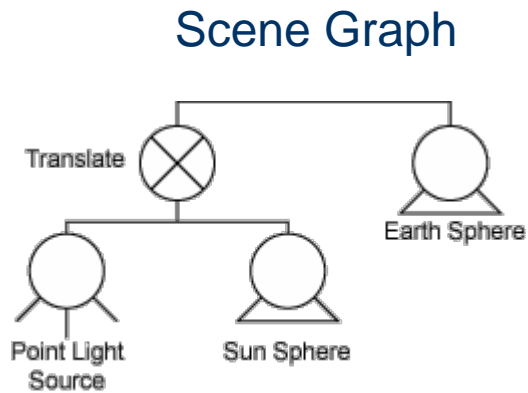


Scene Graph



Lessons: Using Point Light Source

Inline the Earth model with a sun model in a new file



```
#VRML V2.0 utf8
Inline
{
  url " day3_15_EarthWithMaterialNode.wrl "
}
Transform {
  translation 3 2 0
  children [
    PointLight
    {
      ambientIntensity 1
    }
    Inline
    {
      url "sun.wrl"
    }
  ]
}
```

Program day4_1_SunAndEarthWithLight.wrl

- With Node Communication, you can add behavior and interaction to the virtual world (Ch. 8).
 - Objects in the scene can move
 - Object's appearance can change
 - Object can start and stop audio clips and animated textures
- Transform Node control's position, size, and orientation of an object
 - These properties can be changed while your world is being explored... but how?
 - Using Class Data Types and ROUTES

Class	Description
eventIn	An event received by the node
eventOut	An event sent by the node
exposedField	A public node member which can be changed
field	A private node member which can NOT be changed

- Data Types in Transform Node from Appendix A

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

- Routes are the wiring that make animation and user interaction possible.
- Here Node1 is connected to Node2

```
{  
    ... VRML Code  
    ROUTE Node1.isActive TO Node2.set_on  
}
```

- What mechanism activates a ROUTE?

- **Definition:** a *sensor* is a node that sends events out based on user interaction with the scene. One such sensor in VRML is:
 - **TouchSensor:** Reacts when the user touches or drags an object in the world.

- TouchSensor Node Default Fields
 - Most commonly adjusted fields
 - See Appendix A for full definition

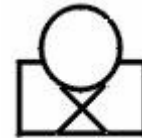
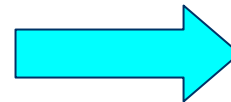
TouchSensor {

enabled TRUE

isActive

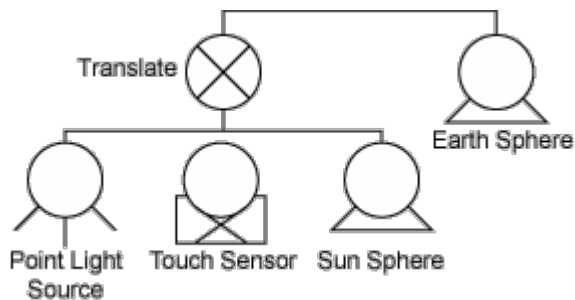
isOver

}



Lessons: Using TouchSensor Node

Scene Graph



```
#VRML V2.0 utf8
Inline
{
  url "day3_15_EarthWithMaterialNode.wrl"
}
Transform
{
  translation 3 2 0
  children [
    DEF SunTS TouchSensor {}
    DEF SunLight PointLight {
      on FALSE
      ambientIntensity 1
    }
    Inline
    {
      url "sun.wrl"
    }
  ]
  ROUTE SunTS.isOver TO SunLight.on
}
```

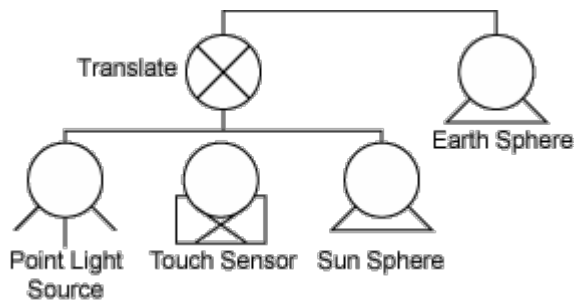
Program day4_2_TouchSensor.wrl

Lessons: Using TouchSensor Node

isOver = "Mouse Is Over"

isActive = "Mouse has Clicked"

Scene Graph



```
#VRML V2.0 utf8
Inline
{
  url "day3_15_EarthWithMaterialNode.wrl"
}
Transform
{
  translation 3 2 0
  children [
    DEF SunTS TouchSensor {}
    DEF SunLight PointLight {
      on FALSE
      ambientIntensity 1
    }
    Inline
    {
      url "sun.wrl"
    }
  ]
  ROUTE SunTS.isActive TO SunLight.on
}
```

Program day4_3_TouchSensorIsActive.wrl

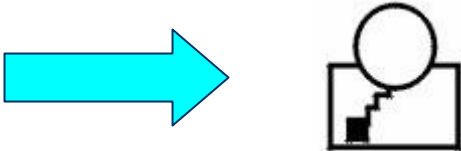
- **Recap:** a *sensor* is a node that sends events out based on user interaction with the scene. We've seen TouchSensors.
 - **TimeSensor:** Indicate the passage of time.

- Definition: The TimeSensor node generates events as time passes. It can be used for
 - continuous simulations and animations
 - periodic activities (for example, one per minute)
 - single occurrence events, such as an alarm clock.

- TimeSensor Node Default Fields
 - Most commonly adjusted fields
 - See Appendix A for full definition

```
TimeSensor {  
  cycleInterval 1  
  enabled TRUE  
  loop FALSE  
  fraction_changed  
}
```

$$\text{fraction_changed} = \frac{\text{Time elapsed}}{\text{cycleInterval}}$$



- Definition: The PositionInterpolator node linearly interpolates among a set of x,y,z values. This is useful for smoothly transitioning between two positions in a transformation.
 - Works in conjunction with a TimeSensor

- PositionInterpolator Node Default Fields
 - Most commonly adjusted fields
 - See Appendix A for full definition

```
PositionInterpolator {  
    key []  
    keyValue []  
    value_changed  
}
```

key	keyValue
0	0 0 0
0.25	0 3 0
0.5	3 3 0
0.75	3 0 0
1	0 0 0

What if input is between key values?

Lessons: Using TouchSensor Node

```
#VRML V2.0 utf8
# Animation clock
DEF Clock TimeSensor
{
  cycleInterval 10.0
  loop TRUE
}
# Animation path
DEF CubePath PositionInterpolator
{
  key
  [
    0.00, 0.25, 0.5, 0.75, 1.0
  ]
  keyValue
  [
    0 0 0, 0 3 0, 3 3 0, 3 0 0, 0 0 0
  ]
}
```

```
...continued
DEF Cube Transform
{
  children Shape
  {
    appearance Appearance
    {
      material Material
      {
        diffuseColor 1 0 0
      }
    }
    geometry Box
    {
      size 1.0 1.0 1.0
    }
  }
}
ROUTE Clock.fraction_changed TO CubePath.set_fraction
ROUTE CubePath.value_changed TO Cube.set_translation
```

- First get sample programs from today's lesson and play
- Add a light to your model from yesterday
- Make objects in the scene rotate or revolve around other objects using the position interpolator in a square-like orbit. Don't worry if it is not a circular path today, just try to animate your model.
 - Examples: electrons revolving around nucleus, planets rotating, planets revolving around the sun.
- Do both models if you finish early and texture mapping if you want to enhance your model.