

X3D Graphics for Web Authors

Chapter 7

Event Animation

If it ain't moving, it ain't 3D.

Andy van Dam, SIGGRAPH Pioneer, Brown University

Contents

Chapter Overview

Concepts

X3D Nodes and Examples

Chapter Summary

References

Chapter Overview

Overview: Event Animation

ROUTE connections and animation

Animation as scene-graph modification

Event-animation design pattern: 10-step process

Interpolation nodes

- TimeSensor and event timing
- ScalarInterpolator and ColorInterpolator
- OrientationInterpolator, PositionInterpolator, PositionInterpolator2D and NormalInterpolator
- CoordinateInterpolator, CoordinateInterpolator2D

[back to Table of Contents](#)

Concepts

ROUTE connections

ROUTE connection enables the output field of one node to pass a value that then stimulates the input field of another node

- The passed value also includes a time stamp

Field data type and accessType must both match between node/field of source and target

- Chapter 1, Technical Introduction lists field types
- Also provided in tooltips and specification
- Authors usually must carefully check these

Animation as scene-graph modification

Behavior = changing a field value in a node,
somewhere in the scene graph

Event = time-stamped value going over a ROUTE

Event cascade is a series of events, each one
triggering the next

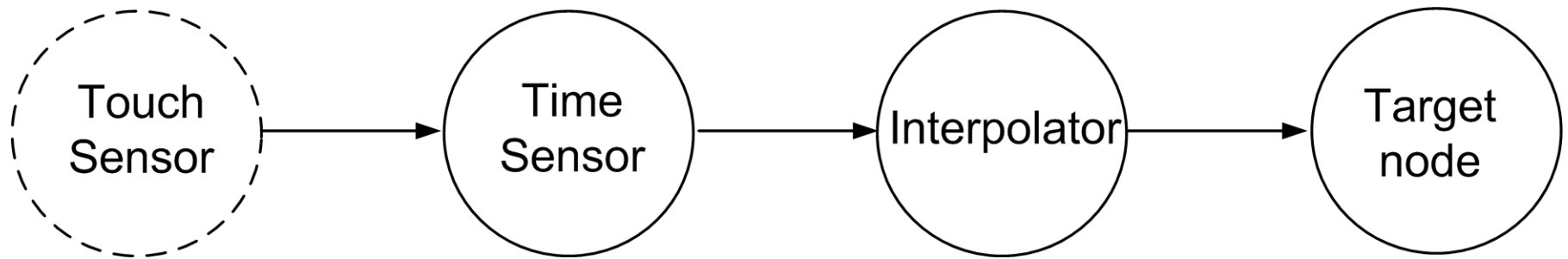
- No event loops allowed, guaranteeing completion

Thus all X3D animation can be considered as
modification of the scene graph at run time

Event-animation design pattern

X3D can be imposing, there are many nodes

Nevertheless a simple design pattern is used for nearly every kind of animation



This consistent event ROUTE pattern enables you to expertly animate most X3D scene behaviors

Review

Field data types

X3D is a strongly typed language

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

Types are either single or multiple-value

- Example: SFFloat, SFVec2f, SFVec3f, SFOrientation

Also have arrays for all types

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

Failure to match data types correctly is an error!

- During schema validation, loading or at run time

X3D has strong data typing

Data typing is very important to prevent errors

- *Strong data typing* means that all data types must match (or be converted) exactly
- *Weak data typing* means data types may be promoted or changed by the system automatically without author direction (or quality control)

Data type errors lead to erroneous computations and system crashes, in any computer language

X3D has strong data typing

- Cost: authors must ensure their scene is correct
- Benefit: mysterious run-time errors avoided

Field data types

| Field-type names | Description | Example values |
|------------------|--|--|
| SFBool | Single-field boolean value | true or false (X3D syntax), TRUE or FALSE (ClassicVRML syntax) |
| MFBool | Multiple-field boolean array | true false false true (X3D syntax), [TRUE FALSE FALSE TRUE] (ClassicVRML syntax) |
| SFColor | Single-field color value, red-green-blue | 0 0.5 1.0 |
| MFColor | Multiple-field color array, red-green-blue | 1 0 0, 0 1 0, 0 0 1 |
| SFColorRGBA | Single-field color value, red-green-blue alpha (opacity) | 0 0.5 1.0 0.75 |
| MFColorRGBA | Multiple-field color array, red-green-blue alpha (opacity) | 1 0 0 0.25, 0 1 0 0.5, 0 0 1 0.75 (red green blue, varying opacity) |
| SFInt32 | Single-field 32-bit integer value | 0 |
| MFInt32 | Multiple-field 32-bit integer array | 1 2 3 4 5 |
| SFFloat | Single-field single-precision floating-point value | 1.0 |
| MFFloat | Multiple-field single-precision floating-point array | -1 2.0 3.14159 |

Field data types

| Field-type names | Description | Example values |
|------------------|---|---|
| SFDouble | Single-field double-precision floating-point value | 2.7128 |
| MFDouble | Multiple-field double-precision array | −1 2.0 3.14159 |
| SFImage | Single-field image value | Contains special pixel-encoding values, see Chapter 5 for details |
| MFImage | Multiple-field image value | Contains special pixel-encoding values, see Chapter 5 for details |
| SFNode | Single-field node | <Shape/> or Shape {space} |
| MFNode | Multiple-field node array of peers | <Shape/><Group/><Transform/> |
| SFRotation | Single-field rotation value using 3-tuple axis, radian angle form | 0 1 0 1.57 |
| MFRotation | Multiple-field rotation array | 0 1 0 0, 0 1 0 1.57, 0 1 0 3.14 |
| SFString | Single-field string value | "Hello world!" |
| MFString | Multiple-field string array | "EXAMINE" "FLY" "WALK" "ANY" |
| SFTime | Single-field time value | 0 |
| MFTime | Multiple-field time array | −1 0 1 567890 |

Field data types

| Field-type names | Description | Example values |
|------------------|--|------------------------|
| SFVec2f/SFVec2d | Single-field 2-float/2-double vector value | 0 1.5 |
| MFVec2f/MFVec2d | Multiple-field 2-float/2-double vector array | 1 0, 2 2, 3 4, 5 5 |
| SFVec3f/SFVec3d | Single-field vector value of 3-float/ 3-double values | 0 1.5 2 |
| MFVec3f/MFVec3d | Multiple-field vector array of 3-float/ 3-double values | 10 20 30, 4.4 –5.5 6.6 |

ClassicVRML syntax notes

- TRUE and FALSE (rather than XML true and false)
- MF multiple-field array values are surrounded by square brackets, e.g. [10 20 30, 4.4 –5.5 6.6]
- No special XML escape characters such as **&**;

accessType: input, output, initialize

accessType determines if field is data sender, receiver, or holder

- inputOnly: can only receive events
- outputOnly: can only send events
- initializeOnly: cannot send or receive
- inputOutput: can send, receive and be initialized

Failure to match accessType correctly is an error!

- Detected during authoring-tool checks, or run time

accessType naming conventions 1

The accessType names were changed when VRML97 was upgraded to X3D

- Functionality remains essentially unchanged

X3D specification entries for each node use yet another shorthand, as shown here

| VRML97 Name | X3D Name | X3D Specification abbreviation |
|--|----------------|--------------------------------|
| eventIn | inputOnly | [in] |
| eventOut | outputOnly | [out] |
| field | initializeOnly | [] |
| exposedField | inputOutput | [in,out] |
| VRML, Virtual reality modeling language; X3D, Extensible 3D. | | |

accessType naming conventions 2

Field names often reveal special accessType

- Prefix *set_* indicates inputOnly field
- Prefix *_changed* indicates outputOnly field
- Prefix *is* for outputOnly boolean field (e.g. isActive)

inputOnly, outputOnly fields not allowed in files

Understanding naming conventions helps authors understand ROUTE definitions and results

Looking ahead: we will name our own fields when creating Scripts and prototypes, further underscoring importance of naming

Interpolating animation chains: 10-step design process

The following 10-step process can be used for all animation tasks

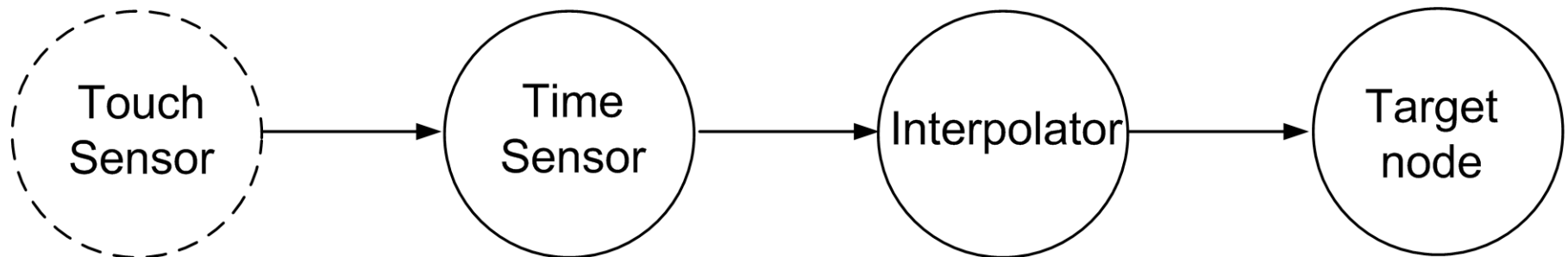
Table is also provided in order to look up how to produce typed-value outputs corresponding to each interpolator or sequencer node

A detailed example follows

This 10-step process is a good check to perform each time you create an animation chain

Interpolating animation chains 1-2

1. ***Pick target.*** Pick node and target field to animate (i.e., field that receives changing animation values)



2. ***Name target.*** Provide a DEF label for the node of interest, giving it a name

Interpolating animation chains 3-4

3. ***Check `accessType` and data type.***

- Ensure target field has *accessType* of `inputOnly` or `inputOutput`, so that it can receive input events
- Determine if target field has floating-point type: `SFFloat`, `SFVec3f`, `MFVec3f`, `SFColor`, and so on...
If so, use an interpolator node as the event source

4. ***Determine whether `Sequencer` or `Script`.***

- If the target type is an `SFBool` or `SFInt32`, use a `sequencer` node as event source
- If the target type is an `SFNode` or `MFNode`, use a `Script` node as the event source

Interpolating animation chains 5-6

5. ***Determine which Interpolator.*** If you are not using a sequencer or Script node, determine corresponding Interpolator which produces the appropriate data type for *value_changed* output using lookup table
 - Example: PositionInterpolator produces SFVec3f *value_changed* events
6. ***Triggering sensor.*** If desired, add sensor node at beginning, to provide appropriate SFTIME or SFBool trigger to start animation
 - Sometimes the triggering event is an output event from another animation chain

Interpolating animation chains 7-8

7. ***TimeSensor clock.*** Add a TimeSensor as the animation clock, then set its *cycleInterval* field to the desired duration interval of animation
 - Set *loop*='false' if an animation only runs once at certain specific times.
 - Set *loop*='true' if it loops repeatedly
8. ***Connect trigger.*** ROUTE sensor or trigger node's output field to the TimeSensor input in order to start the animation chain

Interpolating animation chains 9-10

9. ***Connect clock.*** ROUTE the TimeSensor *fraction_changed* field to the interpolator (or sequencer) node's *set_fraction* field in order to drive the animation chain
10. ***Connect animation output.*** ROUTE the interpolator, sequencer, or Script node's *value_changed* field to target field of interest in order to complete the animation chain

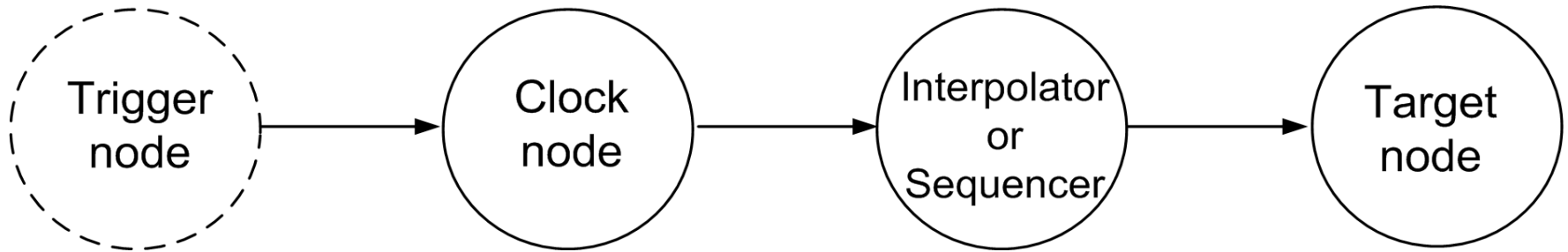
Construction of animation-chain design pattern is now complete, test whether animation works

X3D field types and corresponding animation nodes

| Field type | Description | Interpolator/Sequencer animation nodes |
|------------|---|--|
| SFBool | Single-field boolean value | BooleanSequencer |
| SFColor | Single-field Color value, red-green-blue | ColorInterpolator |
| SFInt32 | Single-field 32-bit Integer value | IntegerSequencer |
| SFFloat | Single-field single-precision floating-point value | ScalarInterpolator |
| SFRotation | Single-field Rotation value using 3-tuple axis, radian angle form | ColorInterpolator |
| SFTime | Single-field Time value | TimeSensor |
| SFVec2f | Single-field 2-float vector value | PositionInterpolator2D |
| MFVec2f | Multiple-field 2-float vector array | CoordinateInterpolator2D |
| SFVec3f | Single-field vector value of 3-float values | PositionInterpolator |
| MFVec3f | Multiple-field vector array of 3-float values | CoordinateInterpolator |

Example animation chains

Each row shows commonly authored sequences of nodes in animation chains

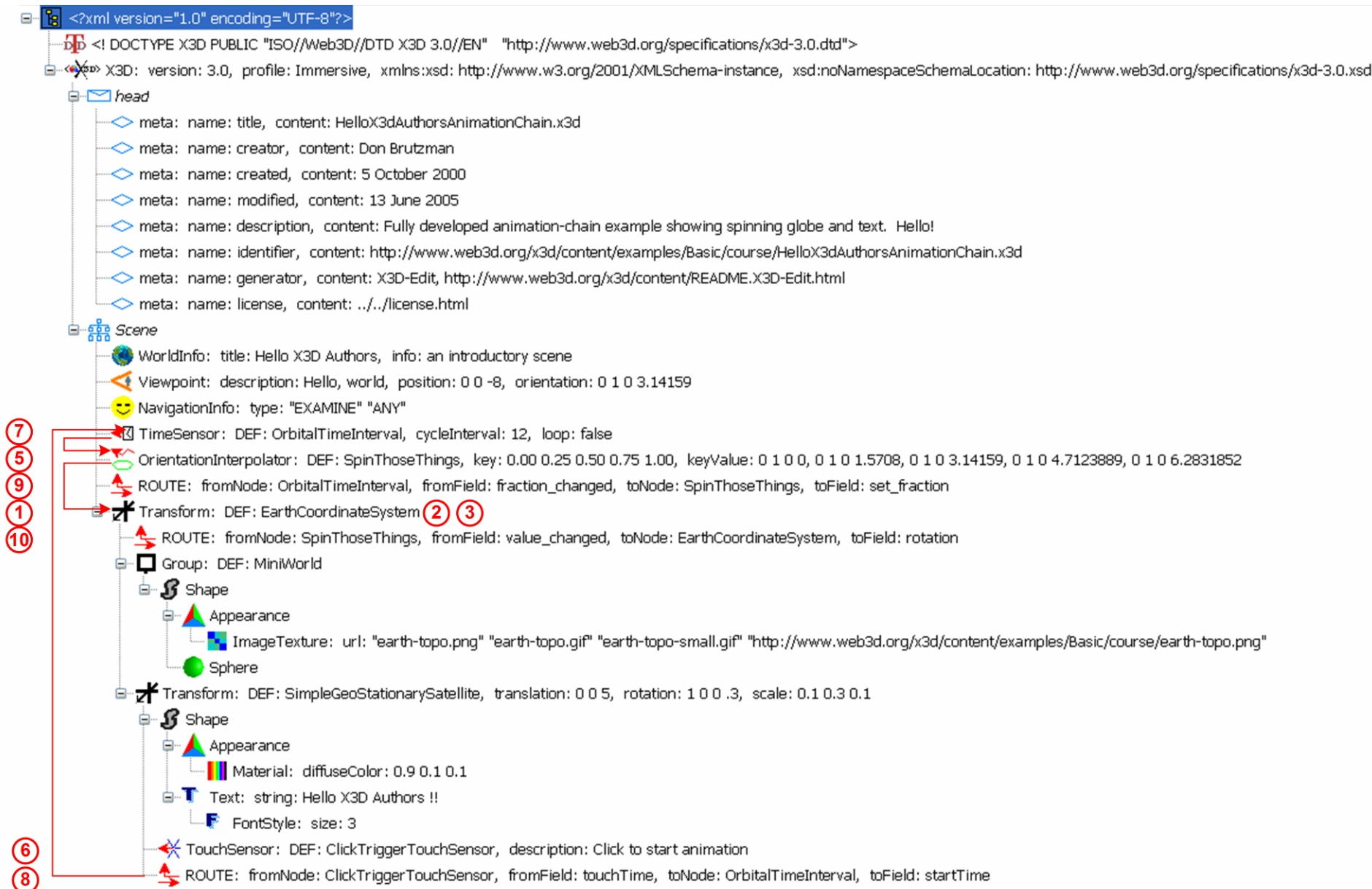


| Triggering Nodes (Optional) | Clock Nodes | Value-Producing Nodes | Value-Consuming Nodes, Fields |
|------------------------------|-------------|-------------------------|--------------------------------|
| TouchSensor | TimeSensor | ScalarInterpolator | Material (transparency) |
| VisibilitySensor | TimeSensor | ColorInterpolator | Material (color field) |
| | TimeSensor | PositionInterpolator | Transform (translation, scale) |
| PrimarySensor | TimeSensor | OrientationInterpolator | Transform (rotation) |
| TouchSensor | | MovieTexture | |
| MovieTexture (loop complete) | TimeSensor | PositionInterpolator2D | Rectangle2D |

Hello X3D Authors showing ROUTEs



Hello X3D Authors 10-step process



Hello X3D Authors 10-step process

- 1. Pick target.** The target node is a Transform, and the target field is *set_rotation*.
- 2. Name target.** The Transform is named *DEF='EarthCoordinateSystem'*.
- 3. Check accessType and data type.** As shown by the Transform node field-definition table in Chapter 3 and the X3D-Edit tooltip, the *set_rotation* field has type SFOrientation.
- 4. Determine whether Sequencer or Script.** These special node types are not applicable to this example, because the data type for *set_rotation* is SFOrientation which is a floating-point type.
- 5. Determine which Interpolator.** The animating OrientationInterpolator is named *DEF="SpinThoseThings"* and placed just before the Transform.
- 6. Triggering sensor.** A triggering TouchSensor is added next to the geometry to be clicked, and then named *DEF='ClickTriggerTouchSensor'*.
- 7. TimeSensor clock.** The TimeSensor is added at the beginning of the chain, named *DEF='OrbitalTimeInterval'* and has both the *cycleInterval* and *loop* fields set.
- 8. Connect trigger.** Add ROUTE to connect the triggering TouchSensor node's *touchTime* output field to the clock node's *startTime* input field.
- 9. Connect clock.** Add ROUTE to connect the clock node's *fraction_changed* output field to the interpolator node's *set_fraction* input field.
- 10. Connect animation output.** Add ROUTE to connect the interpolator node's *value_changed* output field to the original target input field, *set_rotation*.

Interpolation node type

X3DInterpolationNode is the formal name for the interpolation node type

Each interpolation node includes the following common fields and naming conventions

- SF, MF <type> definition must be consistent for node in order to properly define response function

| Type | accessType | Name | Default | Range | Profile |
|---------------|-------------|---------------|---------|---------------------|-------------|
| MFFloat | inputOutput | key | [] | $(-\infty, \infty)$ | Interchange |
| MF<type> | inputOutput | keyValue | [] | (type dependent) | Interchange |
| SFFloat | inputOnly | set_fraction | | | Interchange |
| [SF MF]<type> | outputOnly | value_changed | | | Interchange |
| SFNode | inputOutput | metadata | NULL | [X3DMetadataObject] | Core |

Common interpolator fields

- *key*, *keyValue* hold the point values defining the characteristic function
- *key* array always has type MFFloat
- *keyValue* array data type matches the named type of the parent Interpolator node
 - final value must equal first value in *keyValue* array if smooth looping is desired
- Lengths of *key*, *keyValue* arrays must be equal
- Note that *keyValue* array can hold values which are themselves MF (multi-field) array type
- Function output *value_changed* always has same name, but data type matches the Interpolator node

Interpolation

Interpolation is the estimation of intermediate values from other values

Computing averages is computationally efficient and highly optimizable

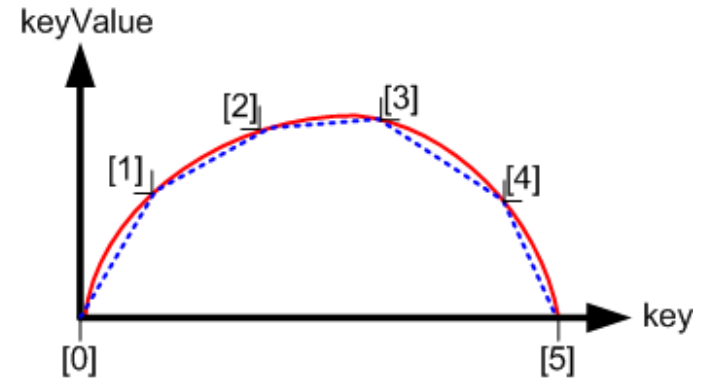
Linear approximation is thus well suited for high-performance graphics animation

X3D provides interpolation nodes for each of the floating-point data types

- including multiple-value types: Color, Vec3f, etc.

Linear interpolation

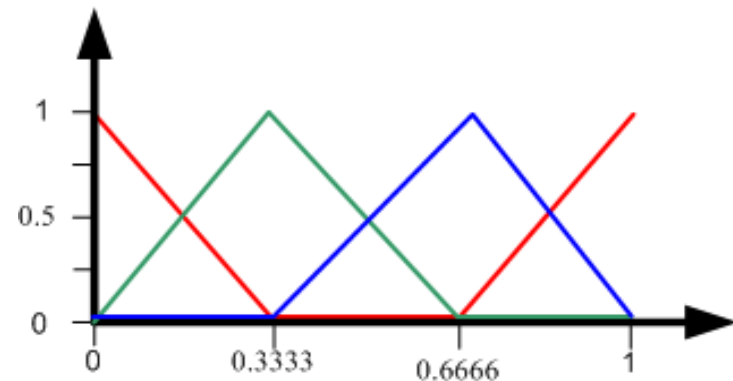
Piecewise-linear curve fitting
can approximate any curve
with arbitrary accuracy



Multi-field (MF) values are
individually interpolated
proportionately

key='0 0.3333 0.666 1'

keyValue='1 0 0, 0 1 0,
0 0 1, 1 0 0'

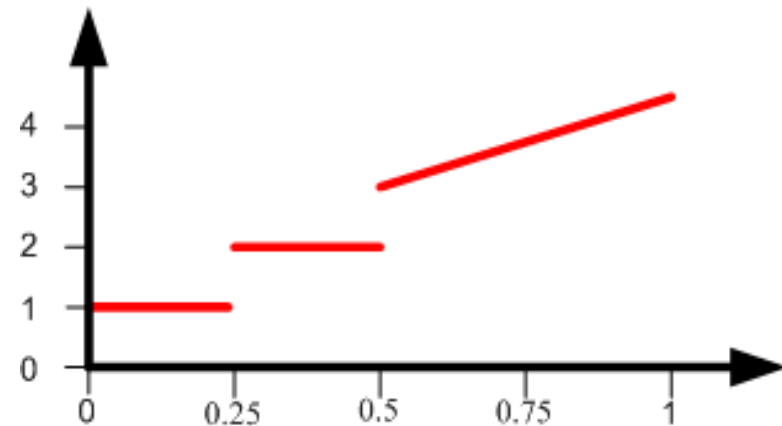


Step-wise linear interpolation

Step functions are created
by repeating time values
and corresponding output

key='0 0.25 0.25 0.5 0.5 1'

keyValue='1 1 2 2 3 4'

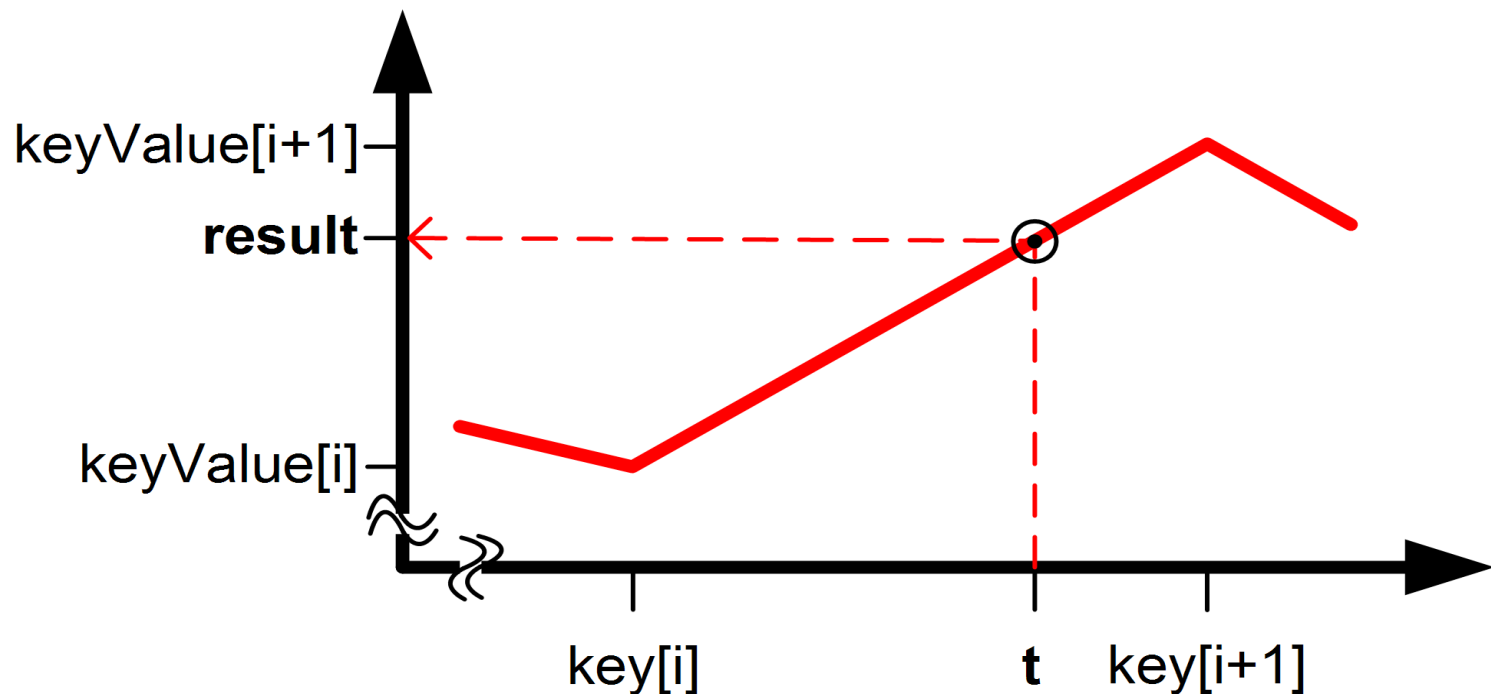


Note that time-fraction key
array must always be
monotonically increasing

Double linear-interpolation averaging

Matched *key*, *keyValue* arrays define the points for a linear-interpolator approximation function

Two-way weighted averaging is used to compute interpolated-input, interpolated-output results



X3D Nodes and Examples

TimeSensor

TimeSensor is the heartbeat of an animation

- provides pulse that triggers event cascades
- initiates computations for drawing next frame

TimeSensor tracks elapsed time based on the computer clock, rather than screen update rate

- Ensures that animations are smooth and realistic
- Fixed (constant) frame rate is typically not feasible since computation varies for screen-image updates

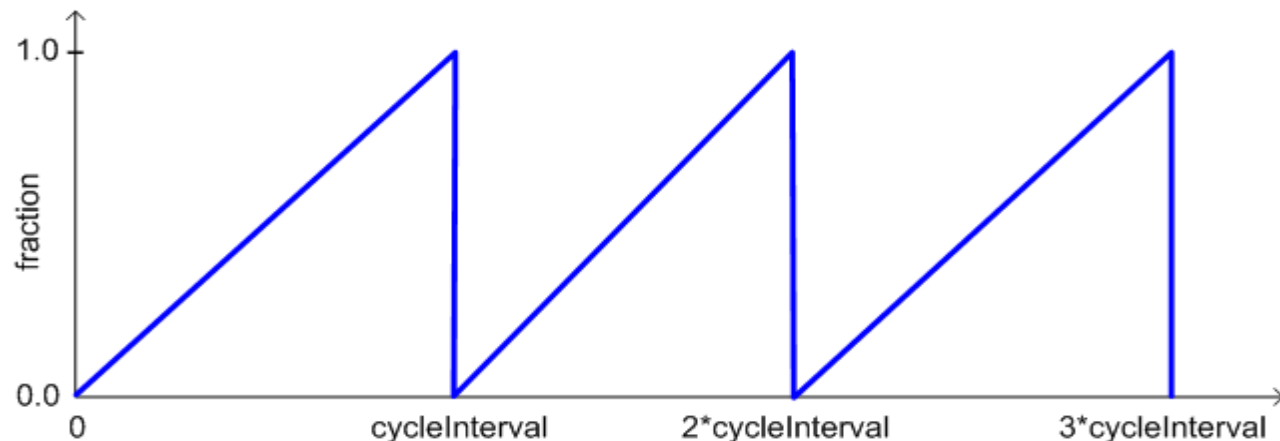
TimeSensor output

Output time is an SFTIME ramp function ranging [0,1] that repeats every *cycleInterval* seconds

- Sometimes called a 'sawtooth' function
- SFFloat output field *fraction_changed* used as input to other interpolators, sequencers

```
time = now
temp = (now - startTime) / cycleInterval
f = fractionalPart (temp)

if (now ≤ startTime)
    fraction_changed = 0.0
if ((f == 0.0) && (now > startTime))
    fraction_changed = 1.0
else fraction_changed = f
```

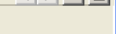


TimeSensor fields 1

- *enabled* controls whether node enabled or disabled
- *loop* is an SFBool indicating whether to continue looping indefinitely after first cycle is complete
- *cycleInterval* defines total loop duration in seconds, either for single-shot animation or looped repetition
- *startTime*, *stopTime* are provided (or contain) SFTIME values for when to start, stop respectively
 - ROUTE an SFTIME value to *set_startTime* or *set_stopTime*
- *cycleTime* field is sent an SFTIME output value upon completion of each loop

TimeSensor fields 2

- *isActive*, *isPaused* are output SFBool true/false events sent whenever the TimeSensor is set to run or paused
- *pauseTime*, *resumeTime* are SFTIME values for current clock time whenever paused or resumed
 - Corresponding boolean *isPaused* event is also sent, with value of true when paused and false when resuming
- *elapsedTime* output provides cumulative number of seconds since TimeSensor was activated and began running, without including paused time



Edit TimeSensor

DEF ☒ AnimationClock

USE ☐ AnimationClock

containerField

☐ children

cycleInterval 6

startTime 0

stopTime 0


pauseTime 0

resumeTime 0

enabled ☒

loop ☐

OK Cancel Help

| | |
|---|---|
|  TimeSensor | TimeSensor continuously generates events as time passes. Typical use: ROUTE thisTimeSensor.fraction_changed TO someInterpolator.set_fraction. Interchange profile hint: TimeSensor may be ignored if cycleInterval < 0.01 second. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| enabled | [enabled: accessType inputOutput, type SFBool (true false) "true"] Enables/disables node operation. |
| cycleInterval | [cycleInterval: accessType inputOutput, type SFTIME CDATA "1.0"] cycleInterval is loop duration in seconds. Interchange profile hint: TimeSensor may be ignored if cycleInterval < 0.01 second. |
| loop | [loop: accessType inputOutput, type SFBool (true false) "false"] Repeat indefinitely when loop=true, repeat only once when loop=false. |
| startTime | [startTime: accessType inputOutput, type SFTIME CDATA "0"] When time now >= startTime, isActive becomes true and TimeSensor becomes active. Absolute time: number of seconds since Jan 1, 1970, 00:00:00 GMT. Hint: usually receives a ROUTED time value. |
| stopTime | [stopTime: accessType inputOutput, type SFTIME CDATA "0"] When stopTime becomes <= time now, isActive becomes false and TimeSensor becomes inactive. Absolute time: number of seconds since Jan 1, 1970, 00:00:00 GMT. Hint: usually receives a ROUTED time value. |
| cycleTime | [cycleTime: accessType outputOnly, type SFTIME CDATA #FIXED ""] cycleTime sends a time outputOnly at startTime, and also at the beginning of each new cycle (useful for synchronization with other time-based objects). |
| isActive | [isActive: accessType outputOnly, type SFBool (true false) #FIXED ""] isActive true/false events are sent when TimeSensor starts/stops running. |
| isPaused | [isPaused: accessType outputOnly, type SFBool (true false) #FIXED ""] isPaused true/false events are sent when TimeSensor is paused/resumed. Warning: not supported in VRML97. |
| pauseTime | [pauseTime: accessType inputOutput, type SFTIME CDATA "0"] When time now >= pauseTime, isPaused becomes true and TimeSensor becomes paused. Absolute time: number of seconds since Jan 1, 1970, 00:00:00 GMT. Hint: usually receives a ROUTED time value. Warning: not supported in VRML97. |
| resumeTime | [resumeTime: accessType inputOutput, type SFTIME CDATA "0"] When resumeTime becomes <= time now, isPaused becomes false and TimeSensor becomes inactive. Absolute time: number of seconds since Jan 1, 1970, 00:00:00 GMT. Hint: usually receives a ROUTED time value. Warning: not supported in VRML97. |
| elapsedTime | [elapsedTime: accessType outputOnly, type SFTIME CDATA #FIXED ""] Current elapsed time since TimeSensor activated/running, cumulative in seconds, and not counting any paused time. Warning: not supported in VRML97. |

| | |
|------------------|--|
| fraction_changed | <p>[fraction_changed: accessType outputOnly, type SFFloat CDATA #FIXED ""]</p> <p>fraction_changed continuously sends value in range [0,1] showing time progress in the current cycle.</p> |
| time | <p>[time: accessType outputOnly, type SFTIME CDATA #FIXED ""]</p> <p>Time continuously sends the absolute time (since January 1, 1970) for a given simulation tick.</p> |
| containerField | <p>[containerField: NMToken "children"]</p> <p>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.</p> |
| class | <p>[class CDATA #IMPLIED]</p> <p>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.</p> |

ScalarInterpolator node

Generates a scalar (single-valued) SFFloat for *value_changed* output

key and *keyValue* arrays contain SFFloat values

set_fraction determines input value to piece-wise linear function

- Percentage between bracketing *key[i]*, *key[i+1]* values used to compute corresponding output *value_changed* as weighted average between *keyValue[i]*, *keyValue[i+1]*
- Which is same algorithm for all interpolators

ScalarInterpolatorExample.x3d - Editor

ScalarInterpolatorExample.x3d

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.1.xsd'>

<head>
<meta content='ScalarInterpolatorExample.x3d' name='title'/>
<meta content='Demonstrate use of ScalarInterpolator to animate transparency.' name='description'/>
<meta content='Don Brutzman' name='creator'/>
<meta content='28 January 2008' name='created'/>
<meta content='28 January 2008' name='modified'/>
<meta content='http://X3dGraphics.com' name='reference'/>
<meta content='http://www.web3d.org/x3d/content/examples/help.html' name='reference'/>
<meta content='Copyright Don Brutzman and Leonard Daly 2007' name='rights'/>
<meta content='X3D book, X3D graphics, X3D-Edit, http://www.x3dgraphics.com' name='subject'/>
<meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter07-EventAnimationInterpolation/ScalarInterpolatorExample.x3d' name='identifier'/>
<meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
<meta content='../license.html' name='license'/>
</head>
<Scene>
<Transform translation='0 -1 0'>
<Shape>
<Sphere radius='2'/>
<Appearance>
<Material DEF='SphereMaterial' diffuseColor='0.941176 0.027451 0' transparency='0'/>
</Appearance>
</Shape>
</Transform>
<TimeSensor DEF='AnimationClock' cycleInterval='8' loop='true'/>
<!-- note that final value equals first value in keyValue array in order to support smooth looping -->
<ScalarInterpolator DEF='TransparencyAnimator' key='0 0.5 1' keyValue='0 1 0'/>
<ROUTE fromField='fraction_changed' fromNode='AnimationClock' toField='set_fraction' toNode='TransparencyAnimator'/>
<ROUTE fromField='value_changed' fromNode='TransparencyAnimator' toField='set_transparency' toNode='SphereMaterial'/>
<!-- notice that Text appears later in scene although it is located above Sphere -->
<Transform translation='0 1.5 0'>
<Shape>
<Text string='"Animating transparency" "using ScalarInterpolator"'>
<FontStyle justify='"MIDDLE" "MIDDLE"'/>
</Text>
</Shape>
</Transform>
</Scene>
</X3D>

X3D Viewer

X3D Viewer

Animating transparency
using ScalarInterpolator

Animating transparency
using ScalarInterpolator

30:24

INS

Edit ScalarInterpolator

DEF ☒ TransparencyAnima

USE ☐ parencyAnimator

containerField

☐ children

key, keyValue arrays

| key | keyValue |
|-----|----------|
| 0 | 0 |
| 0.5 | 1 |
| 1 | 0 |


+

-

OK

Cancel

Help

| | |
|---|--|
|  ScalarInterpolator | ScalarInterpolator generates piecewise-linear values that can be ROUTED to other Float attributes. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keys must match number of keyValues! |
| keyValue | [keyValue: accessType inputOutput, type MFFloat CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keys must match number of keyValues! |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type SFFloat CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

ColorInterpolator node

Generates a 3-tuple (triple-valued) SFColor for continuous *value_changed* output

key array contains SFFloat values

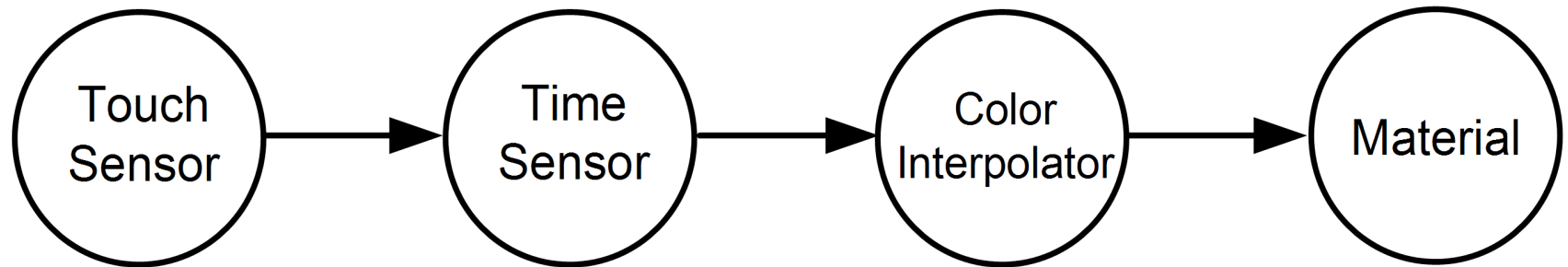
keyValue array contains SFColor values

Linear interpolation of red, green, blue (RGB) values is respectively performed for each bracketing *keyValue* pair

ColorInterpolator animation chain

Each node's output field matches data type of next node's input field

accessType outputOnly to inputOnly, initializeOnly also match



TextTriggerTouchSensor

output: touchTime

AnimationClock

input: startTime
output: fraction_changed

ColorChanger

input: set_fraction
output: value_changed

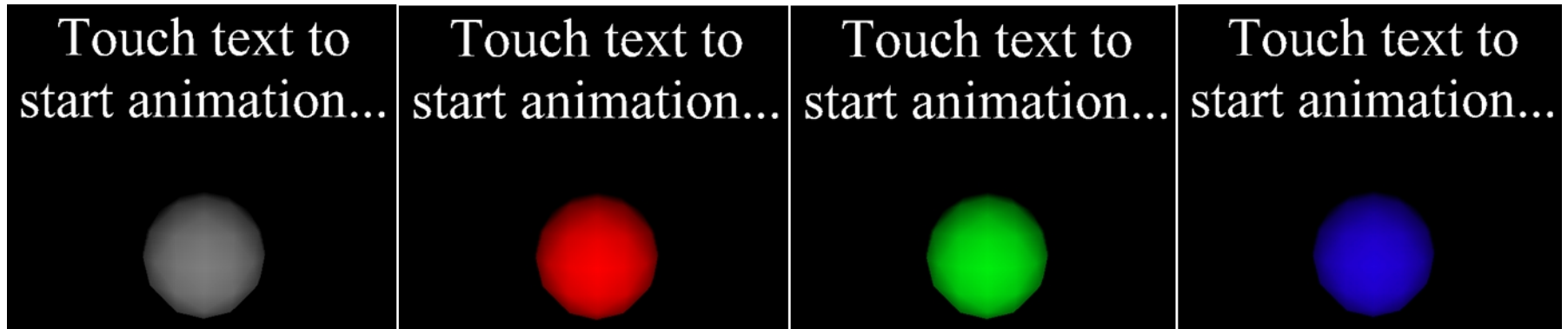
SphereMaterial

input: diffuseColor

ColorInterpolator example output

Using the pointing device to select the text triggers the ColorInterpolator animation

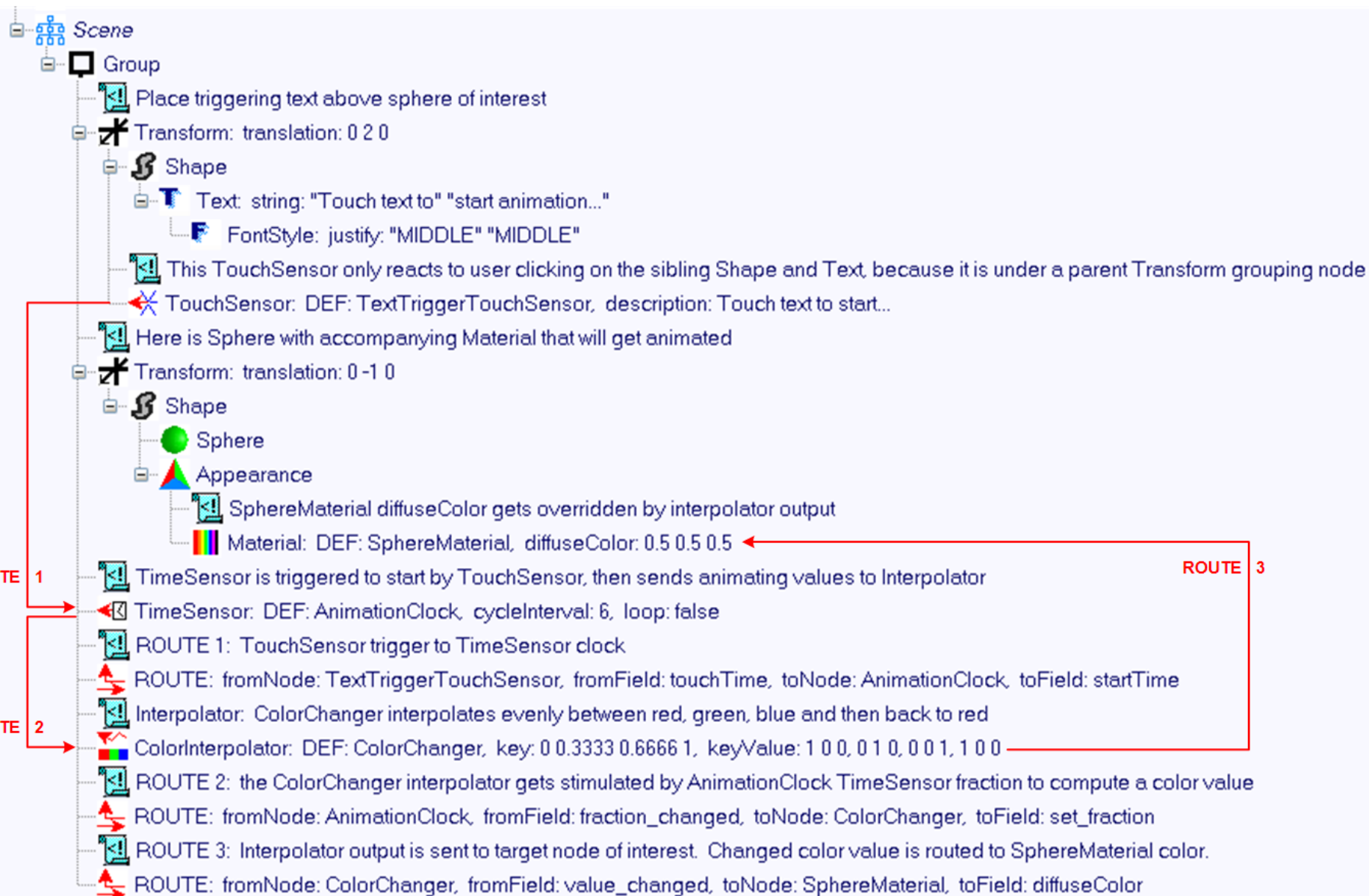
- Colors vary by linear interpolation of component red-green-blue RGB values



ColorInterpolator scene graph illustration



ColorInterpolator scene graph with ROUTEs



```

<Group>
  <!-- Place triggering text above sphere of interest -->
  <Transform translation='0 2 0'>
    <Shape>
      <Text string='"Touch text to" "start animation...">
        <FontStyle justify="MIDDLE" "MIDDLE"/>
      </Text>
    </Shape>
    <!-- This TouchSensor only reacts to user clicking on the sibling Shape and Text, because it is under a parent Transform grouping node -->
    * — <!-- TextTriggerTouchSensor ROUTE: [from touchTime to AnimationClock.startTime] -->
    <TouchSensor DEF='TextTriggerTouchSensor' description='Touch text to start...'/>
  </Transform>
  <!-- Here is Sphere with accompanying Material that will get animated -->
  <Transform translation='0 -1 0'>
    <Shape>
      <Sphere/>
      <Appearance>
        <!-- SphereMaterial diffuseColor gets overridden by interpolator output -->
        * — <!-- SphereMaterial ROUTE: [from ColorChanger.value changed to diffuseColor] -->
        <Material DEF='SphereMaterial' diffuseColor='0.5 0.5 0.5'/>
      </Appearance>
    </Shape>
  </Transform>
  <!-- TimeSensor is triggered to start by TouchSensor, then sends animating values to Interpolator -->
  * — <!-- AnimationClock ROUTEs: [from TextTriggerTouchSensor.touchTime to startTime] [from fraction changed to ColorChanger.set_fraction] -->
  <TimeSensor DEF='AnimationClock' cycleInterval='6'/>
  <!-- ROUTE 1: TouchSensor trigger to TimeSensor clock -->
  <ROUTE fromNode='TextTriggerTouchSensor' fromField='touchTime' toNode='AnimationClock' toField='startTime'/>
  <!-- Interpolator: ColorChanger interpolates evenly between red, green, blue and then back to red -->
  * — <!-- ColorChanger ROUTEs: [from AnimationClock.fraction changed to set_fraction] [from value changed to SphereMaterial.diffuseColor] -->
  <ColorInterpolator DEF='ColorChanger' key='0 0.3333 0.6666 1' keyValue='1 0 0, 0 1 0, 0 0 1, 1 0 0'/>
  <!-- ROUTE 2: the ColorChanger interpolator gets stimulated by AnimationClock TimeSensor fraction to compute a color value -->
  <ROUTE fromNode='AnimationClock' fromField='fraction_changed' toNode='ColorChanger' toField='set_fraction'/>
  <!-- ROUTE 3: Interpolator output is sent to target node of interest. Changed color value is routed to SphereMaterial color. -->
  <ROUTE fromNode='ColorChanger' fromField='value_changed' toNode='SphereMaterial' toField='diffuseColor'/>
</Group>

```

* — indicates autogenerated comments showing incoming, outgoing events

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
3 <X3D profile='Immersive' version='3.0' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
4     xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.1.xsd'>
5
6 <head>
7 <meta content='ColorInterpolatorExample.x3d' name='filename' />
8 <meta content='Demonstrate basic design pattern for animating a node.' name='description' />
9 <meta content='Don Brutzman' name='creator' />
10 <meta content='17 April 2005' name='created' />
11 <meta content='27 January 2008' name='modified' />
12 <meta content='ColorInterpolatorExampleSceneGraphWithRoutes.png' name='drawing' />
13 <meta content='Animation ColorInterpolator' name='keywords' />
14 <meta content='Gapos;http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter07-EventAnimationInterpolation/ColorInterpolatorExample.x3d' name='identifier' />
15 <meta content='X3D-Edit, http://www.web3d.org/x3d/content/README.X3D-Edit.html' name='generator' />
16 <meta content='../.../license.html' name='license' />
17 </head>
18 <Scene>
19 <Group>
20 <!-- Place triggering text above sphere of interest -->
21 <Transform translation='0 2 0'>
22 <Shape>
23 <Text string='Touch text to "start animation..."'
24 <FontStyle justify='MIDDLE' MIDDLE' />
25 </Text>
26 </Shape>
27 <!-- This TouchSensor only reacts to user clicking on the sibling Shape and Text,
28 because it is under a parent Transform grouping node -->
29 <TouchSensor DEF='TextTriggerTouchSensor' description='Touch text to start...' />
30 </Transform>
31 <!-- Here is Sphere with accompanying Material that will get animated -->
32 <Transform translation='0 -1 0'>
33 <Shape>
34 <Sphere />
35 <Appearance>
36 <!-- SphereMaterial diffuseColor gets overridden by interpolator output -->
37 <Material DEF='SphereMaterial' diffuseColor='0.5 0.5 0.5' />
38 </Appearance>
39 </Shape>
40 </Transform>
41 <!-- TimeSensor is triggered to start by TouchSensor, then sends animating values to Interpolator -->
42 <TimeSensor DEF='AnimationClock' cycleInterval='6' loop='false' />
43 <!-- ROUTE 1: TouchSensor trigger to TimeSensor clock -->
44 <ROUTE fromField='touchTime' fromNode='TextTriggerTouchSensor' toField='startTime' toNode='AnimationClock' />
45 <!-- Interpolator: ColorChanger interpolates evenly between red, green, blue and then back to red -->
46 <ColorInterpolator DEF='ColorChanger' key='0 0.3333 0.6666 1' keyValue='1 0 0 1 0 0 0 1 1 0 0' />
47 <!-- ROUTE 2: the ColorChanger interpolator gets stimulated by AnimationClock TimeSensor fraction to compute a color value -->
48 <ROUTE fromField='fraction_changed' fromNode='AnimationClock' toField='set_fraction' toNode='ColorChanger' />
49 <!-- ROUTE 3: Interpolator output is sent to target node of interest. Changed color value is routed to SphereMaterial color. -->
50 <ROUTE fromField='value_changed' fromNode='ColorChanger' toField='diffuseColor' toNode='SphereMaterial' />
51 </Group>
52 </Scene>
53 </X3D>

```

Edit ColorInterpolator

DEF ☒ ColorChanger


USE ☐ ColorChanger

keys/keyValues

| key | r | g | b | ... |
|--------|---|---|---|-------|
| 0 | 1 | 0 | 0 | Red |
| 0.3333 | 0 | 1 | 0 | Green |
| 0.6666 | 0 | 0 | 1 | Blue |
| 1 | 1 | 0 | 0 | Red |

+ -

OK Cancel Help

| | |
|---|---|
|  ColorInterpolator | ColorInterpolator generates a range of Color values that can be ROUTED to a <Color> node's color attribute. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction. Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keys must match number of keyValues! |
| keyValue | [keyValue: accessType inputOutput, type MFColor CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keys must match number of keyValues! |
| set_fraction | [set_fraction: accessType inputOnly, type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type SFColor CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

OrientationInterpolator node

Generates a 4-tuple (four-valued orientation)
SFRotation for *value_changed* output

key array contains SFFloat values

keyValue array contains SFRotation values

- As always: same number of *key*, *keyValue* entries

OrientationInterpolator animates along shortest path between the two normal vectors, also computes linear average between two corresponding angles, in *keyValue* array

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
  xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.1.xsd'>

  <head>
    <meta content='PositionOrientationInterpolatorsExample.x3d' name='title' />
    <meta content='Demonstrate use of PositionInterpolator and OrientationInterpolator to animate object motion.'
      name='description' />
    <meta content='Don Brutzman' name='creator' />
    <meta content='29 January 2008' name='created' />
    <meta content='29 January 2008' name='modified' />
    <meta content='http://X3dGraphics.com' name='reference' />
    <meta content='http://www.web3d.org/x3d/content/examples/help.html' name='reference' />
    <meta content='Copyright Don Brutzman and Leonard Daly 2007' name='rights' />
    <meta content='X3D book, X3D graphics, X3D-Edit, http://www.x3dgraphics.com' name='subject' />
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter07-EventAnimationInterpolation/PositionOrientationInterpolatorsExample.x3d' name='identifier' />
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator' />
    <meta content='../license.html' name='license' />
  </head>
  <Scene>
    <Viewpoint description='Animation demo' orientation='1 0 1 -0.2' position='0 4 10' />
    <Transform DEF='Pointer' translation='1 0 1'>
      <Transform rotation='1 0 0 1.57'>
        <Shape>
          <Cone bottomRadius='0.5' height='1.5' />
          <Appearance>
            <Material DEF='ConeMaterial' diffuseColor='0.427451 1 0.160784' />
          </Appearance>
        </Shape>
      </Transform>
    </Transform>
    <Shape DEF='Floor'>
      <Box size='10 0.05 10' />
      <Appearance>
        <Material diffuseColor='0 0.262745 0.941176' />
      </Appearance>
    </Shape>
    <TimeSensor DEF='AnimationClock' cycleInterval='10' loop='true' />
    <!-- note that final value equals first value in keyValue array in order to support smooth looping -->
    <!-- first drive around the location -->
    <PositionInterpolator DEF='PositionAnimator' key='0 0.2 0.25 0.45 0.5 0.7 0.75 0.95 1'
      keyValue='-4 0 -4 -4 0 4 -4 0 4 4 0 4 4 0 -4 4 0 -4 -4 0 -4 -4 0 -4' />
    <ROUTE fromField='fraction_changed' fromNode='AnimationClock' toField='set_fraction' toNode='PositionAnimator' />
    <ROUTE fromField='value_changed' fromNode='PositionAnimator' toField='set_translation' toNode='Pointer' />
    <!-- then rotate the pointer to match next direction while paused at each position -->
    <OrientationInterpolator DEF='OrientationAnimator' key='0 0.2 0.25 0.45 0.5 0.7 0.75 0.95 1'
      keyValue='0 1 0 0 1 0 0 0 1 0 1.57 0 1 0 1.57 0 1 0 3.14 0 1 0 3.14 0 1 0 4.71 0 1 0 4.71 0 1 0 6.283' />
    <!-- final rotation value is 2pi rather than 0 so that rotation animation is smooth, not flip-flopping -->
    <ROUTE fromField='fraction_changed' fromNode='AnimationClock' toField='set_fraction' toNode='OrientationAnimator' />
    <ROUTE fromField='value_changed' fromNode='OrientationAnimator' toField='set_rotation' toNode='Pointer' />
    <!-- notice that explanatory Text appears later in scene although it is located above driving plane -->
    <Transform translation='0 3.5 0'>
      <Shape>
        <Text string='Animation using PositionInterpolator' and OrientationInterpolator
          <FontStyle justify='MIDDLE' 'MIDDLE' size='7' />
        </Text>
      </Shape>
    </Transform>
  </Scene>
</X3D>

```

Edit OrientationInterpolator

DEF ☒ OrientationAnimator containerField

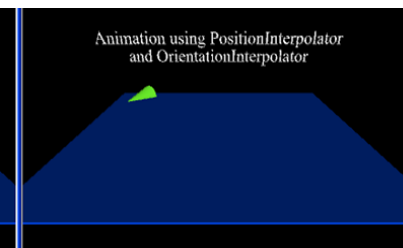
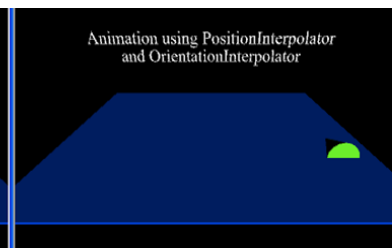
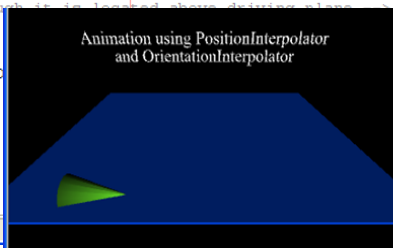
USE ☐ entationAnimator ☐ children


key, keyValue arrays

| key | axis-x | axis-y | axis-z | angle |
|------|--------|--------|--------|-------|
| 0 | 0 | 1 | 0 | 0 |
| 0.2 | 0 | 1 | 0 | 0 |
| 0.25 | 0 | 1 | 0 | 1.57 |
| 0.45 | 0 | 1 | 0 | 1.57 |
| 0.5 | 0 | 1 | 0 | 3.14 |
| 0.7 | 0 | 1 | 0 | 3.14 |
| 0.75 | 0 | 1 | 0 | 4.71 |
| 0.95 | 0 | 1 | 0 | 4.71 |
| 1 | 0 | 1 | 0 | 6.283 |

+ -

OK Cancel Help



| | |
|---|---|
|  OrientationInterpolator | <p>OrientationInterpolator generates a series of rotation values Results can be ROUTED to a <Transform> node's 'rotation' attribute or another Rotations attribute Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute.</p> |
| DEF | <p>[DEF ID #IMPLIED]</p> <p>DEF defines a unique ID name for this node, referencable by other nodes.</p> <p>Hint: descriptive DEF names improve clarity and help document a model.</p> |
| USE | <p>[USE IDREF #IMPLIED]</p> <p>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.</p> <p>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.</p> <p>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute!</p> |
| key | <p>[key: accessType inputOutput, type MFFloat CDATA #IMPLIED]</p> <p>Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues.</p> <p>Hint: number of keys must match number of keyValues!</p> |
| keyValue | <p>[keyValue: accessType inputOutput, type MFRotation CDATA #IMPLIED]</p> <p>Output values for linear interpolation, each corresponding to time-fraction keys.</p> <p>Hint: number of keys must match number of keyValues!</p> |
| set_fraction | <p>[set_fraction: inputOnly type SFFloat CDATA #FIXED ""]</p> <p>set_fraction selects input key for corresponding keyValue output.</p> |
| value_changed | <p>[value_changed: accessType outputOnly, type SFRotation CDATA #FIXED ""]</p> <p>Linearly interpolated output value determined by current key time and corresponding keyValue pair.</p> |
| containerField | <p>[containerField: NMTOKEN "children"]</p> <p>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes.</p> |
| class | <p>[class CDATA #IMPLIED]</p> <p>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes.</p> |

PositionInterpolator node

Generates a 3-tuple (three-valued floating point) SFVec3f for *value_changed* output

key array contains SFFloat values

keyValue array contains SFVec3f values

- As always: same number of *key*, *keyValue* entries

PositionInterpolator computes weighted average between corresponding x, y and z pairs in the *keyValue* array

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
    xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.1.xsd'>

<head>
  <meta content='PositionOrientationInterpolatorsExample.x3d' name='title' />
  <meta content='Demonstrate use of PositionInterpolator and OrientationInterpolator to animate object motion.' name='description' />
  <meta content='Don Brutzman' name='creator' />
  <meta content='29 January 2008' name='created' />
  <meta content='29 January 2008' name='modified' />
  <meta content='http://X3dGraphics.com' name='reference' />
  <meta content='http://www.web3d.org/x3d/content/examples/help.html' name='reference' />
  <meta content='Copyright Don Brutzman and Leonard Daly 2007' name='rights' />
  <meta content='X3D book, X3D graphics, X3D-Edit, http://www.x3dgraphics.com' name='subject' />
  <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter07-EventAnimationInterpolation/PositionOrientationInterpolatorsExample.x3d'
    name='identifier' />
  <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator' />
  <meta content='../license.html' name='license' />
</head>
<Scene><Viewpoint description="Animation demo" position="0 4 10" orientation="1 0 1 -0.2"/>
  <Transform translation='1 0 1' DEF='Pointer'>
    <Transform rotation='1 0 0 1.57'>
      <Shape>
        <Cone bottomRadius="0.5" height="1.5"/>
        <Appearance><Material DEF="ConeMaterial" diffuseColor="0.427451 1 0.160784"/></Appearance>
      </Shape>
    </Transform>
  </Transform>
  <Shape DEF="Floor">
    <Box size="10 0.05 10"/>
    <Appearance><Material diffuseColor="0 0.262745 0.941176"/></Appearance>
  </Shape>
  <TimeSensor DEF='AnimationClock' cycleInterval='10' loop='true' />
  <!-- note that final value equals first value in keyValue array in order to support smooth looping -->
  <!-- first drive around the location -->
  <PositionInterpolator DEF="PositionAnimator" key="0 0.2 0.25 0.45 0.5 0.7 0.75 0.95 1"
    keyValue="-4 0 -4, -4 0 4, -4 0 4, 4 0 4, 4 0 -4, 4 0 -4, -4 0 -4, -4 0 -4"/>
  <ROUTE fromField='fraction_changed' fromNode='AnimationClock' toField='set_fraction' toNode='PositionAnimator' />
  <ROUTE fromField='value_changed' fromNode='PositionAnimator' toField='set_translation' toNode='Pointer' />
  <!-- then rotate the pointer to match next direction while paused at each position -->
  <OrientationInterpolator DEF="OrientationAnimator" key="0 0.2 0.25 0.45 0.5 0.7 0.75 0.95 1"
    keyValue="0 1 0 0, 0 1 0 0, 0 1 0 1.57, 0 1 0 1.57, 0 1 0 3.14, 0 1 0 3.14, 0 1 0 4.71, 0 1 0 4.71, 0 1 0 6.283"/>
  <!-- final rotation value is 2pi rather than 0 so that rotation animation is smooth, not flip-flopping -->
  <ROUTE fromField='fraction_changed' fromNode='AnimationClock' toField='set_fraction' toNode='OrientationAnimator' />
  <ROUTE fromField='value_changed' fromNode='OrientationAnimator' toField='set_rotation' toNode='Pointer' />
  <!-- notice that explanatory Text appears later in scene although it is located above driving plane -->
  <Transform translation='0 3.5 0'>
    <Shape>
      <Text string='Animation using PositionInterpolator and OrientationInterpolator'
        FontStyle justify='MIDDLE' MIDDLE size='.7' />
    </Shape>
  </Transform>
</Scene>
</X3D>

```

Edit PositionInterpolator

DEF ☒ PositionAnimator
 USE ☐ PositionAnimator

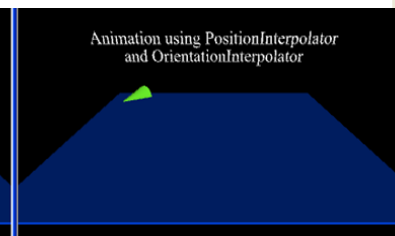
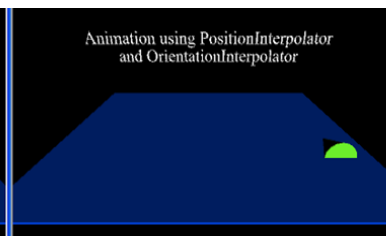
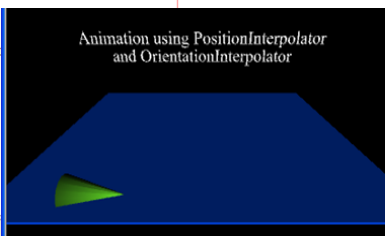
containerField
☐ children


key, keyValue arrays

| key | x | y | z |
|------|----|---|----|
| 0 | -4 | 0 | -4 |
| 0.2 | -4 | 0 | 4 |
| 0.25 | -4 | 0 | 4 |
| 0.45 | 4 | 0 | 4 |
| 0.5 | 4 | 0 | 4 |
| 0.7 | 4 | 0 | -4 |
| 0.75 | 4 | 0 | -4 |
| 0.95 | -4 | 0 | -4 |
| 1 | -4 | 0 | -4 |

+ -

OK Cancel Help



| | |
|---|---|
|  PositionInterpolator | PositionInterpolator generates a series of triplet values. Results can be ROUTED to a <Transform> node's 'translation' attribute or another Vector3Float attribute Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keys must match number of keyValues! |
| keyValue | [keyValue: accessType inputOutput, type MFVec3f CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keys must match number of keyValues! |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type SFVec3f CDATA #FIXED "";] Linearly interpolated output value determined by current key time and corresponding keyValue pair. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

PositionInterpolator2D node

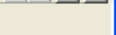
Generates a 2-tuple (two-valued floating point) SFVec2f for *value_changed* output

key array contains SFFloat values

keyValue array contains SFVec2f values

- As always: same number of *key*, *keyValue* entries

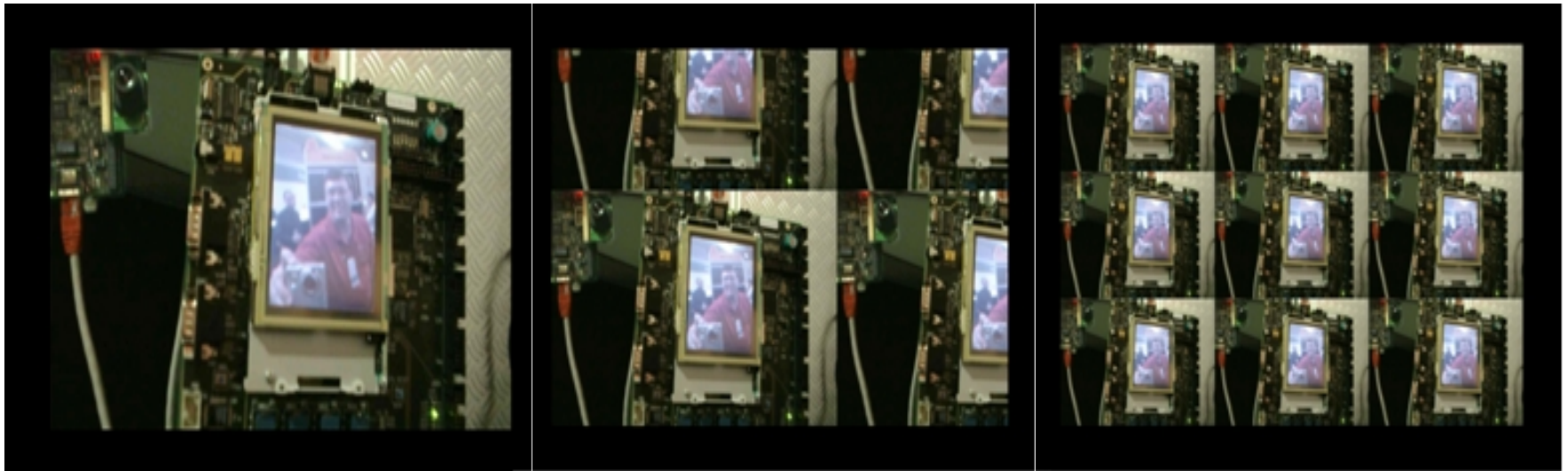
PositionInterpolator computes weighted average between corresponding x and y pairs in the *keyValue* array




A small LCD screen is mounted on a green printed circuit board (PCB). The screen displays a video of a person with dark hair wearing a red shirt, sitting at a table. The PCB has various electronic components, including a green LED, capacitors, and a USB cable plugged into a port on the left. The background is a white textured surface.

23:28

PositionInterpolator2D screen captures



Selecting the texture with the mouse pointer starts the TextureTransform *scale* animation, deselecting the texture stops the animation

| | |
|--|--|
|  PositionInterpolator2D | PositionInterpolator2D generates a series of Vector2Float values that can be ROUTED to a Vector2Float attribute. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction. Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| keyValue | [keyValue: accessType inputOutput, type MFVec2f CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type SFVec2f CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

NormalInterpolator node

Generates a 3-tuple (three-valued floating point) SFVec3f for *value_changed* output

key array contains SFFloat values


keyValue array contains SFVec3f values

- As always: same number of *key*, *keyValue* entries
- SFVec3f outputs: unit-normal vectors, magnitude=1

NormalInterpolator animates along shortest path between the two normal vectors currently being referenced in *keyValue* array

NormalInterpolator node X3D-Edit

[TODO: example needed]

| | |
|--|--|
|  NormalInterpolator | NormalInterpolator generates a series of normal (perpendicular) vector sets along the surface of a unit sphere ROUTE values to vector attribute of a <Normal> node or another Vector3FloatArray attribute Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keys must match number of keyValues! |
| keyValue | [keyValue: accessType inputOutput, type MFVec3f CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keys must match number of keyValues! |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type MFVec3f CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

CoordinateInterpolator node

Generates n -tuple (multiple-valued floating point) array, MFFloat for *value_changed* output

key array contains n SFFloat values


keyValue array contains n MFFloat values

- As always: same number of *key*, *keyValue* entries
- Counting is very important for arrays of arrays!

CoordinateInterpolator computes weighted average between corresponding element pairs for each subarray in the *keyValue* array

CoordinateInterpolator node X3D-Edit

[TODO: example needed]

| | |
|---|--|
|  CoordinateInterpolator | CoordinateInterpolator generates a series of Coordinate values that can be ROUTED to a <Coordinate> node's 'point' attribute or another Vector3FloatArray attribute. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction. Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnly. |
| keyValue | [keyValue: accessType inputOutput, type MFVec3f CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnly. |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type MFVec3f CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnly. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

CoordinateInterpolator2D node

Generates 2-tuple (two-valued floating point) array, MFVec2f for *value_changed* output

key array contains SFFloat values

keyValue array contains MFVec2f values

- As always: same number of *key*, *keyValue* entries
- Counting is very important for arrays of arrays!

CoordinateInterpolator2D computes weighted average between corresponding x and y pairs for each subarray in the *keyValue* array

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D profile='Immersive' version='3.1' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
  xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.1.xsd'>

  <head>
    <component level='3' name='Interpolation' />
    <meta content='CoordinateInterpolator2dExample.x3d' name='title' />
    <meta content='Example to interpolate using CoordinateInterpolator2D - click geometry to activate animation loop.' name='description' />
    <meta content='Don Brutzman, Jeff Weekley, Jane Wu' name='creator' />
    <meta content='9 October 2001' name='created' />
    <meta content='30 January 2008' name='modified' />
    <meta content='CoordinateInterpolator2D' name='subject' />
    <meta content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter07-EventAnimationInterpolation/CoordinateInterpolator2dExample.x3d' name='identifier' />
    <meta content='http://www.web3d.org/x3d/content/examples/Basic/development/CoordinateInterpolator2dExample.x3d' name='reference' />
    <meta content='X3D-Edit, http://www.web3d.org/x3d/content/README.X3D-Edit.html' name='generator' />
    <meta content='../license.html' name='license' />
  </head>
  <Scene>
    <Viewpoint description='Click to activate animation' orientation='1 0 0 -0.4' position='0 4 10' />
    <TimeSensor DEF='Clock' cycleInterval='5' enabled='false' loop='true' />
    <CoordinateInterpolator2D DEF='InterpolateCrossSection' key='0 0.45 0.9 1'
      keyValue='1 1 1 -1 -1 -1 1 1 1 2 2 2 -2 -1 -1 -1 1 2 2 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1' />
    <ROUTE fromField='fraction_changed' fromNode='Clock' toField='set_fraction' toNode='InterpolateCrossSection' />
    <Transform translation='0.25 1 0'>
      <!-- &amp; is the XML escape character code for ampersand character -->
      <TouchSensor DEF='Toucher' description='click &amp; hold shape to animate Extrusion' />
      <ROUTE fromField='isActive' fromNode='Toucher' toField='enabled' toNode='Clock' />
      <!-- also reset clock to restart -->
      <ROUTE fromField='touchTime' fromNode='Toucher' toField='startTime' toNode='Clock' />
    <Shape>
      <Appearance>
        <Material diffuseColor='0.2 0.8 0.4' />
      </Appearance>
      <Extrusion DEF='AnimatedCrossSectionExtrusion' crossSection='1 1, 1 -1, -1 -1, -1 1, 1 1'
        spine='-4 0 -2, -1 0 -2, 2 0 1, 2 0 4' />
    </Shape>
    <ROUTE fromField='value_changed' fromNode='InterpolateCrossSection'
      toField='set_crossSection' toNode='AnimatedCrossSectionExtrusion' />
  </Transform>
  <Transform translation='-1.5 -1 2'>
    <Billboard axisOfRotation='0 0 0'>
      <Shape>
        <Text string='click &amp; hold shape "to animate Extrusion"'>
          <FontStyle family='SANS' justify='MIDDLE' MIDDLE' size='0.8' />
        </Text>
        <Appearance>
          <Material diffuseColor='0.8 0.4 0.2' />
        </Appearance>
      </Shape>
    </Billboard>
  </Transform>
</Scene>
</X3D>

```

Edit CoordinateInterpolator2D

DEF ☒ InterpolateCrossSec containerField

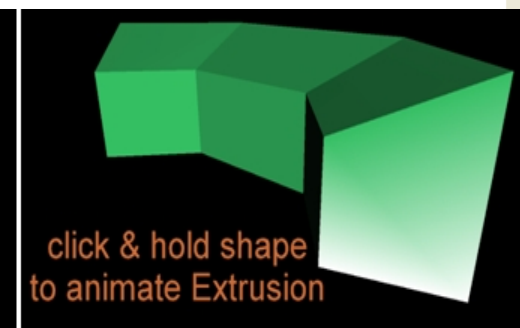
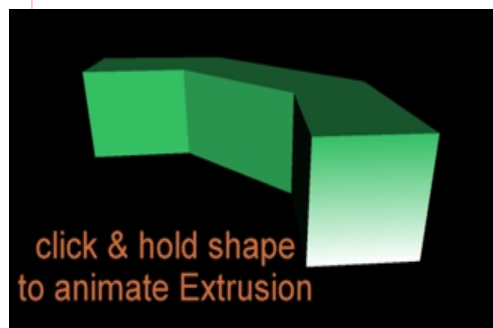
USE ☐ plateCrossSection ☐ children


key, keyValue arrays

| key | x | y |
|------|----|----|
| 0 | 1 | 1 |
| 0.45 | 1 | -1 |
| 0.9 | -1 | -1 |
| 1 | -1 | 1 |

+ -

OK Cancel Help



| | |
|---|--|
|  CoordinateInterpolator2D | CoordinateInterpolator2D generates a series of Vector2FloatArray values that can be ROUTED to a Vector2FloatArray attribute. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction. Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnly. |
| keyValue | [keyValue: accessType inputOutput, type MFVec3f CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnly. |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type MFVec2f CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnly. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

Chapter Summary

Chapter Summary: Event Animation

ROUTE connections and animation

Animation as scene-graph modification

Event-animation design pattern: 10-step process

Interpolation nodes

- TimeSensor and event timing
- ScalarInterpolator and ColorInterpolator
- OrientationInterpolator, PositionInterpolator, PositionInterpolator2D and NormalInterpolator
- CoordinateInterpolator, CoordinateInterpolator2D

References

References 1

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

- Chapter 7, Event Animation
- <http://x3dGraphics.com>
- <http://x3dgraphics.com/examples/X3dForWebAuthors>

X3D Examples Help

- <http://www.web3d.org/x3d/content/examples/help.html>

References 2

X3D Scene Authoring Hints

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

X3D Graphics Specification

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

Pocock, Lynn and Judson Rosebush,
The Computer Animator's Technical Handbook,
Morgan Kaufmann Publishers, 2001.

References 3

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

- <http://www.wiley.com/legacy/compbooks/vrml2sbk/cover/cover.htm>
- <http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook>
- Chapter 08 – Animating Position Orientation Scale

Contact

Don Brutzman

brutzman@nps.edu

<http://web.nps.navy.mil/~brutzman>

Code USW/Br, Naval Postgraduate School
Monterey California 93943-5000 USA

1.831.656.2149 voice

1.831.656.7599 fax

Open-source license

Copyright (c) 1995-2008 held by the author(s). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the names of the Naval Postgraduate School (NPS) Modeling Virtual Environments and Simulation (MOVES) Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

X3D Graphics for Web Authors

Chapter 7

Event Animation

If it ain't moving, it ain't 3D.

Andy van Dam, SIGGRAPH Pioneer, Brown University



Contents

Chapter Overview

Concepts

X3D Nodes and Examples

Chapter Summary

References



Chapter Overview



Overview: Event Animation

ROUTE connections and animation

Animation as scene-graph modification

Event-animation design pattern: 10-step process

Interpolation nodes

- TimeSensor and event timing
- ScalarInterpolator and ColorInterpolator
- OrientationInterpolator, PositionInterpolator, PositionInterpolator2D and NormalInterpolator
- CoordinateInterpolator, CoordinateInterpolator2D

[back to Table of Contents](#)

Concepts



ROUTE connections

ROUTE connection enables the output field of one node to pass a value that then stimulates the input field of another node

- The passed value also includes a time stamp

Field data type and accessType must both match between node/field of source and target

- Chapter 1, Technical Introduction lists field types
- Also provided in tooltips and specification
- Authors usually must carefully check these

Animation as scene-graph modification

Behavior = changing a field value in a node,
somewhere in the scene graph

Event = time-stamped value going over a ROUTE

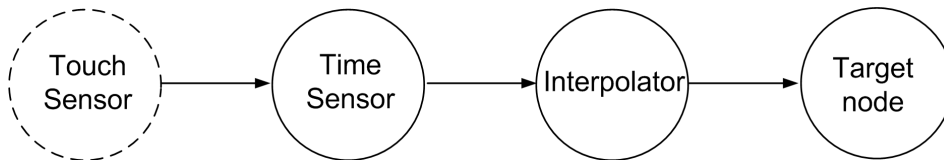
Event cascade is a series of events, each one
triggering the next

- No event loops allowed, guaranteeing completion

Thus all X3D animation can be considered as
modification of the scene graph at run time

Event-animation design pattern

X3D can be imposing, there are many nodes
Nevertheless a simple design pattern is used for
nearly every kind of animation



This consistent event ROUTE pattern enables you
to expertly animate most X3D scene behaviors

web|3D
CONSORTIUM



X3D for Web Authors, Figure 7.1, p. 189.

TouchSensor is optional. Some other triggering event may be provided to start the animation chain, or the TimeSensor may be looping indefinitely.

There are many interpolator nodes. The choice of which interpolator to utilize is determined by the data type of the target field in the target node.

A sequencer node is used instead of an interpolator node if the target field is boolean or integer. Sequencer nodes are described in Chapter 9, Event Utilities and Scripting.

Review

Field data types

X3D is a strongly typed language

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

Types are either single or multiple-value

- Example: SFFloat, SFVec2f, SFVec3f, SFOrientation

Also have arrays for all types

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

Failure to match data types correctly is an error!

- During schema validation, loading or at run time



Data type and accessType information is available for each node in the X3D Tooltips and X3D Specification.

When speaking about data types, you can substitute “array of” for the “MF” prefix.
Example: “MFColor is an array of Color values.”

X3D has strong data typing

Data typing is very important to prevent errors

- *Strong data typing* means that all data types must match (or be converted) exactly
- *Weak data typing* means data types may be promoted or changed by the system automatically without author direction (or quality control)

Data type errors lead to erroneous computations and system crashes, in any computer language

X3D has strong data typing

- Cost: authors must ensure their scene is correct
- Benefit: mysterious run-time errors avoided

web|3D
CONSORTIUM



Strong data typing, XML validation and a number of other X3D quality-control checks prevent the dreaded errors which arise from Garbage In Garbage Out (GIGO).

GIGO errors can be quite difficult to detect, debug and correct. Thus they are best avoided in the first place. Strong typing, XML validation and tools that report errors are an X3D author's best friend.

Field data types

| Field-type names | Description | Example values |
|------------------|--|--|
| SFBool | Single-field boolean value | true or false (X3D syntax), TRUE or FALSE (ClassicVRML syntax) |
| MFBool | Multiple-field boolean array | true false false true (X3D syntax), [TRUE FALSE FALSE TRUE] (ClassicVRML syntax) |
| SFColor | Single-field color value, red-green-blue | 0 0.5 1.0 |
| MFColor | Multiple-field color array, red-green-blue | 1 0 0, 0 1 0, 0 0 1 |
| SFColorRGBA | Single-field color value, red-green-blue alpha (opacity) | 0 0.5 1.0 0.75 |
| MFColorRGBA | Multiple-field color array, red-green- blue alpha (opacity) | 1 0 0 0.25, 0 1 0 0.5, 0 0 1 0.75 (red green blue, varying opacity) |
| SFInt32 | Single-field 32-bit integer value | 0 |
| MFInt32 | Multiple-field 32-bit integer array | 1 2 3 4 5 |
| SFFloat | Single-field single-precision floating- point value | 1.0 |
| MFFloat | Multiple-field single-precision floating- point array | −1 2.0 3.14159 |

X3D for Web Authors, Table 1.4, pp. 19-20.

Field data types

| Field-type names | Description | Example values |
|------------------|---|---|
| SFDouble | Single-field double-precision floating-point value | 2.7128 |
| MFDouble | Multiple-field double-precision array | -1 2.0 3.14159 |
| SFImage | Single-field image value | Contains special pixel-encoding values, see Chapter 5 for details |
| MFImage | Multiple-field image value | Contains special pixel-encoding values, see Chapter 5 for details |
| SFNode | Single-field node | <Shape/> or Shape {space} |
| MFNode | Multiple-field node array of peers | <Shape/><Group/><Transform/> |
| SFRotation | Single-field rotation value using 3-tuple axis, radian angle form | 0 1 0 1.57 |
| MFRotation | Multiple-field rotation array | 0 1 0 0, 0 1 0 1.57, 0 1 0 3.14 |
| SFString | Single-field string value | "Hello world!" |
| MFString | Multiple-field string array | "EXAMINE" "FLY" "WALK" "ANY" |
| SFTime | Single-field time value | 0 |
| MFTime | Multiple-field time array | -1 0 1 567890 |

X3D for Web Authors, Table 1.4, pp. 19-20.

Field data types

| Field-type names | Description | Example values |
|------------------|--|------------------------|
| SFVec2f/SFVec2d | Single-field 2-float/2-double vector value | 0 1.5 |
| MFVec2f/MFVec2d | Multiple-field 2-float/2-double vector array | 1 0, 2 2, 3 4, 5 5 |
| SFVec3f/SFVec3d | Single-field vector value of 3-float/ 3-double values | 0 1.5 2 |
| MFVec3f/MFVec3d | Multiple-field vector array of 3-float/ 3-double values | 10 20 30, 4.4 –5.5 6.6 |

ClassicVRML syntax notes

- TRUE and FALSE (rather than XML true and false)
- MF multiple-field array values are surrounded by square brackets, e.g. [10 20 30, 4.4 –5.5 6.6]
- No special XML escape characters such as **&**;

X3D for Web Authors, Table 1.4, pp. 19-20.

accessType: input, output, initialize

accessType determines if field is data sender, receiver, or holder

- inputOnly: can only receive events
- outputOnly: can only send events
- initializeOnly: cannot send or receive
- inputOutput: can send, receive and be initialized

Failure to match accessType correctly is an error!

- Detected during authoring-tool checks, or run time



Data type and accessType information is available for each node in the X3D Tooltips and X3D Specification.

accessType naming conventions 1

The accessType names were changed when VRML97 was upgraded to X3D

- Functionality remains essentially unchanged

X3D specification entries for each node use yet another shorthand, as shown here

| VRML97 Name | X3D Name | X3D Specification abbreviation |
|--|----------------|--------------------------------|
| eventIn | inputOnly | [in] |
| eventOut | outputOnly | [out] |
| field | initializeOnly | [] |
| exposedField | inputOutput | [in,out] |
| VRML, Virtual reality modeling language; X3D, Extensible 3D. | | |

X3D for Web Authors, Table 1.6, p. 28.

accessType naming conventions 2

Field names often reveal special accessType

- Prefix *set_* indicates inputOnly field
- Prefix *_changed* indicates outputOnly field
- Prefix *is* for outputOnly boolean field (e.g. isActive)

inputOnly, outputOnly fields not allowed in files

Understanding naming conventions helps authors understand ROUTE definitions and results

Looking ahead: we will name our own fields when creating Scripts and prototypes, further underscoring importance of naming

Interpolating animation chains: 10-step design process

The following 10-step process can be used for all animation tasks

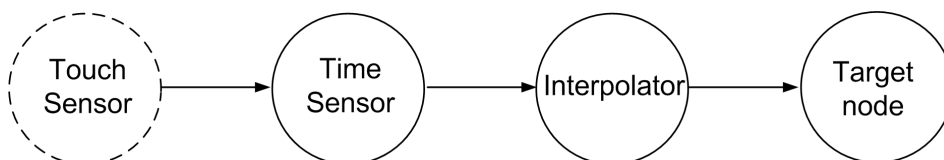
Table is also provided in order to look up how to produce typed-value outputs corresponding to each interpolator or sequencer node

A detailed example follows

This 10-step process is a good check to perform each time you create an animation chain

Interpolating animation chains 1-2

1. **Pick target.** Pick node and target field to animate (i.e., field that receives changing animation values)



2. **Name target.** Provide a DEF label for the node of interest, giving it a name

Interpolating animation chains 3-4

3. ***Check `accessType` and data type.***

- Ensure target field has *accessType* of `inputOnly` or `inputOutput`, so that it can receive input events
- Determine if target field has floating-point type: `SFFloat`, `SFVec3f`, `MFVec3f`, `SFColor`, and so on...
If so, use an interpolator node as the event source

4. ***Determine whether `Sequencer` or `Script`.***

- If the target type is an `SFBool` or `SFInt32`, use a sequencer node as event source
- If the target type is an `SFNode` or `MFNode`, use a Script node as the event source



X3D for Web Authors, section 2.5, pp. 192-193

When checking data type:

- The target field can either be singleton SF type or array MF type
- SF means Single Field, MF means Multiple Field (i.e. an array) in the X3D type-naming convention

Interpolating animation chains 5-6

5. ***Determine which Interpolator.*** If you are not using a sequencer or Script node, determine corresponding Interpolator which produces the appropriate data type for *value_changed* output using lookup table
 - Example: PositionInterpolator produces SFVec3f *value_changed* events
6. ***Triggering sensor.*** If desired, add sensor node at beginning, to provide appropriate SFTIME or SFBool trigger to start animation
 - Sometimes the triggering event is an output event from another animation chain

Interpolating animation chains 7-8

7. ***TimeSensor clock.*** Add a TimeSensor as the animation clock, then set its *cycleInterval* field to the desired duration interval of animation
 - Set *loop*='false' if an animation only runs once at certain specific times.
 - Set *loop*='true' if it loops repeatedly
8. ***Connect trigger.*** ROUTE sensor or trigger node's output field to the TimeSensor input in order to start the animation chain

Interpolating animation chains 9-10

9. **Connect clock.** ROUTE the TimeSensor *fraction_changed* field to the interpolator (or sequencer) node's *set_fraction* field in order to drive the animation chain
10. **Connect animation output.** ROUTE the interpolator, sequencer, or Script node's *value_changed* field to target field of interest in order to complete the animation chain

Construction of animation-chain design pattern is now complete, test whether animation works

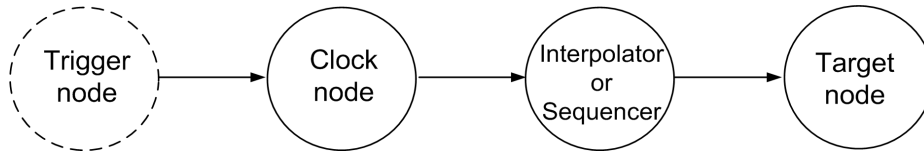
X3D field types and corresponding animation nodes

| Field type | Description | Interpolator/Sequencer animation nodes |
|------------|---|--|
| SFBool | Single-field boolean value | BooleanSequencer |
| SFColor | Single-field Color value, red-green-blue | ColorInterpolator |
| SFInt32 | Single-field 32-bit Integer value | IntegerSequencer |
| SFFloat | Single-field single-precision floating-point value | ScalarInterpolator |
| SFRotation | Single-field Rotation value using 3-tuple axis, radian angle form | ColorInterpolator |
| SFTime | Single-field Time value | TimeSensor |
| SFVec2f | Single-field 2-float vector value | PositionInterpolator2D |
| MFVec2f | Multiple-field 2-float vector array | CoordinateInterpolator2D |
| SFVec3f | Single-field vector value of 3-float values | PositionInterpolator |
| MFVec3f | Multiple-field vector array of 3-float values | CoordinateInterpolator |

X3D for Web Authors, Table 7.5, p. 199.

Example animation chains

Each row shows commonly authored sequences of nodes in animation chains



| Triggering Nodes (Optional) | Clock Nodes | Value-Producing Nodes | Value-Consuming Nodes, Fields |
|------------------------------|-------------|-------------------------|--------------------------------|
| TouchSensor | TimeSensor | ScalarInterpolator | Material (transparency) |
| VisibilitySensor | TimeSensor | ColorInterpolator | Material (color field) |
| | TimeSensor | PositionInterpolator | Transform (translation, scale) |
| PrimarySensor | TimeSensor | OrientationInterpolator | Transform (rotation) |
| TouchSensor | | MovieTexture | |
| MovieTexture (loop complete) | TimeSensor | PositionInterpolator2D | Rectangle2D |

X3D for Web Authors, Table 7.5, p. 199.

Hello X3D Authors showing ROUTEs

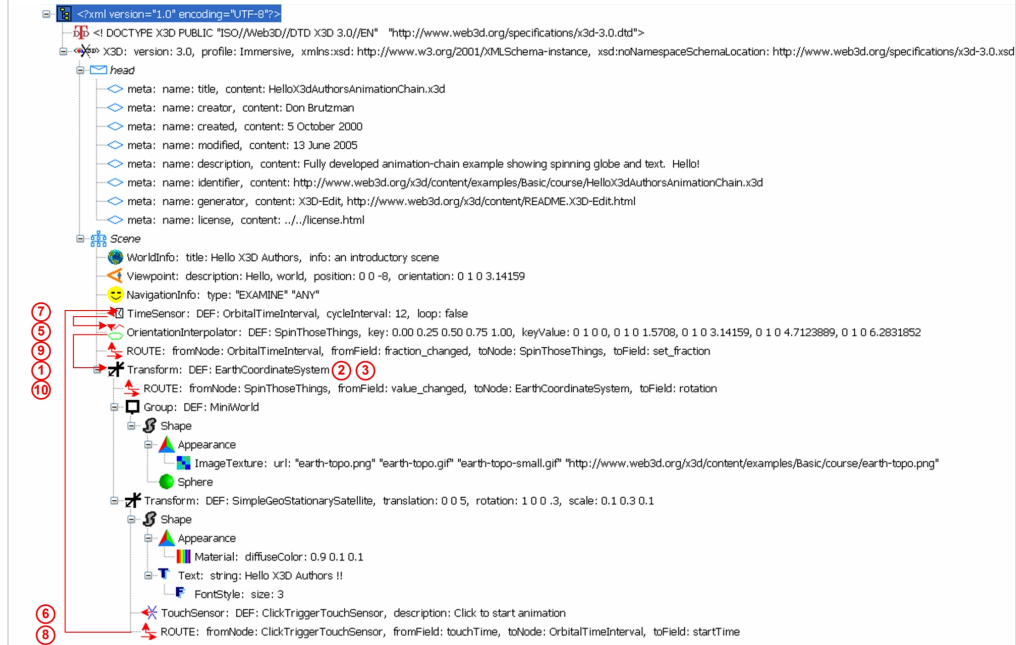


X3D for Web Authors, Figure 7.5, pp. 193-195.

<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter07-EventAnimationInterpolation/HelloX3dAuthorsAnimationChain.x3d>

10-step process for constructing animation chains, applied to animated HelloWorld example

Hello X3D Authors 10-step process



X3D for Web Authors, Figure 7.5, pp. 193-195.

<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter07-EventAnimationInterpolation/HelloX3dAuthorsAnimationChain.x3d>

10-step process for constructing animation chains, applied to animated HelloWorld example

Hello X3D Authors 10-step process

1. **Pick target.** The target node is a Transform, and the target field is *set_rotation*.
2. **Name target.** The Transform is named *DEF='EarthCoordinateSystem'*.
3. **Check accessType and data type.** As shown by the Transform node field-definition table in Chapter 3 and the X3D-Edit tooltip, the *set_rotation* field has type SFOrientation.
4. **Determine whether Sequencer or Script.** These special node types are not applicable to this example, because the data type for *set_rotation* is SFOrientation which is a floating-point type.
5. **Determine which Interpolator.** The animating OrientationInterpolator is named *DEF="SpinThoseThings"* and placed just before the Transform.
6. **Triggering sensor.** A triggering TouchSensor is added next to the geometry to be clicked, and then named *DEF='ClickTriggerTouchSensor'*.
7. **TimeSensor clock.** The TimeSensor is added at the beginning of the chain, named *DEF='OrbitalTimeInterval'* and has both the *cycleInterval* and *loop* fields set.
8. **Connect trigger.** Add ROUTE to connect the triggering TouchSensor node's *touchTime* output field to the clock node's *startTime* input field.
9. **Connect clock.** Add ROUTE to connect the clock node's *fraction_changed* output field to the interpolator node's *set_fraction* input field.
10. **Connect animation output.** Add ROUTE to connect the interpolator node's *value_changed* output field to the original target input field, *set_rotation*.



X3D for Web Authors, Figure 7.5, pp. 193-195.

<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter07-EventAnimationInterpolation/HelloX3dAuthorsAnimationChain.x3d>

10-step process for constructing animation chains, applied to animated HelloWorld example

Interpolation node type

X3DInterpolationNode is the formal name for the interpolation node type

Each interpolation node includes the following common fields and naming conventions

- SF, MF <type> definition must be consistent for node in order to properly define response function

| Type | accessType | Name | Default | Range | Profile |
|---------------|-------------|---------------|---------|---------------------|-------------|
| MFFloat | inputOutput | key | [] | $(-\infty, \infty)$ | Interchange |
| MF<type> | inputOutput | keyValue | [] | (type dependent) | Interchange |
| SFFloat | inputOnly | set_fraction | | | Interchange |
| [SF MF]<type> | outputOnly | value_changed | | | Interchange |
| SFNode | inputOutput | metadata | NULL | [X3DMetadataObject] | Core |

X3D for Web Authors, Table 7.4, p. 197.

Common interpolator fields

- *key*, *keyValue* hold the point values defining the characteristic function
- *key* array always has type MFFloat
- *keyValue* array data type matches the named type of the parent Interpolator node
 - final value must equal first value in *keyValue* array if smooth looping is desired
- Lengths of *key*, *keyValue* arrays must be equal
- Note that *keyValue* array can hold values which are themselves MF (multi-field) array type
- Function output *value_changed* always has same name, but data type matches the Interpolator node

Interpolation

Interpolation is the estimation of intermediate values from other values

Computing averages is computationally efficient and highly optimizable

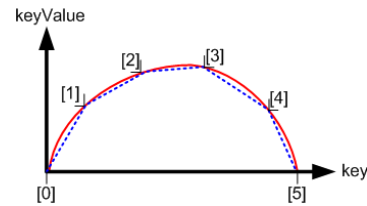
Linear approximation is thus well suited for high-performance graphics animation

X3D provides interpolation nodes for each of the floating-point data types

- including multiple-value types: Color, Vec3f, etc.

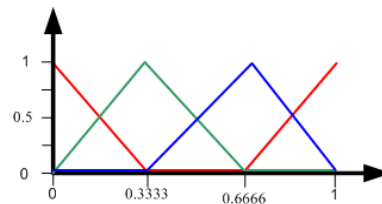
Linear interpolation

Piecewise-linear curve fitting
can approximate any curve
with arbitrary accuracy



Multi-field (MF) values are
individually interpolated
proportionately

key='0 0.3333 0.666 1'
keyValue='1 0 0, 0 1 0,
0 0 1, 1 0 0'



First figure: *X3D for Web Authors*, Figure 7.2, p. 191.

```
<ScalarInterpolator key="0 0.2 0.4 0.6 0.8 1" keyValue="0 5 8 9 4 0"/>
```

Second figure: *X3D for Web Authors*, Figure 7.4, p. 192.

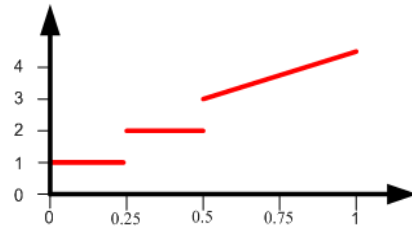
```
<ColorInterpolator key="0, 0.3333, 0.6666, 1" keyValue="1 0 0, 0 1 0, 0 0 1, 1 0 0"/>
```

Step-wise linear interpolation

Step functions are created
by repeating time values
and corresponding output

key='0 0.25 0.25 0.5 0.5 1'

keyValue='1 1 2 2 3 4'



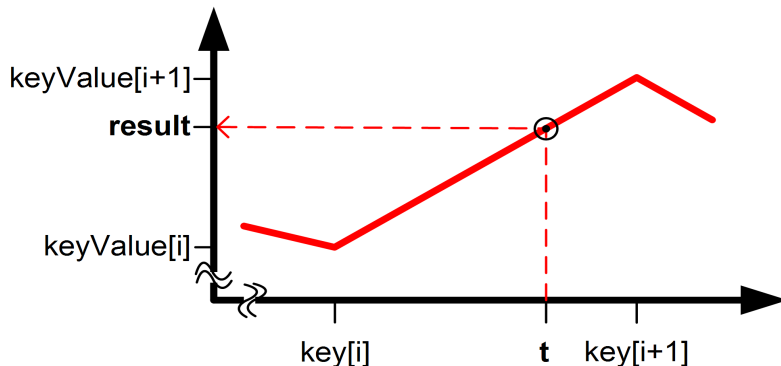
Note that time-fraction key
array must always be
monotonically increasing

X3D for Web Authors, Figure 7.3, p. 191.

<ScalarInterpolator key="0 0.25, 0.25 0.5, 0.5 1" keyValue="1 1, 2 2, 3 4"/>

Double linear-interpolation averaging

Matched *key*, *keyValue* arrays define the points for a linear-interpolator approximation function
Two-way weighted averaging is used to compute interpolated-input, interpolated-output results



X3D for Web Authors, Figure 7.8, p. 198.

First the entry-value t is compared to the *key* array until the prior and following values of *key* are found that are less-than and greater-than t .

Then a percentage is computed that accounts for the proportion of t between the bracketing values of $key[i]$ and $key[i+1]$.

Then this same percentage is applied to compute a new *result* value which equals the same percentage between corresponding output-array values of $keyValue[i]$ and $keyValue[i+1]$.

[back to Table of Contents](#)

X3D Nodes and Examples



TimeSensor

TimeSensor is the heartbeat of an animation

- provides pulse that triggers event cascades
- initiates computations for drawing next frame

TimeSensor tracks elapsed time based on the computer clock, rather than screen update rate

- Ensures that animations are smooth and realistic
- Fixed (constant) frame rate is typically not feasible since computation varies for screen-image updates

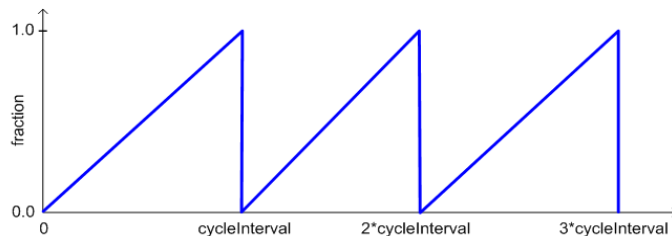
TimeSensor output

Output time is an SFTIME ramp function ranging [0,1] that repeats every *cycleInterval* seconds

- Sometimes called a 'sawtooth' function
- SFFloat output field *fraction_changed* used as input to other interpolators, sequencers

```
time = now
temp = (now - startTime) / cycleInterval
f = fractionalPart (temp)
```

```
if (now ≤ startTime)
    fraction_changed = 0.0
if ((f == 0.0) && (now > startTime))
    fraction_changed = 1.0
else fraction_changed = f
```

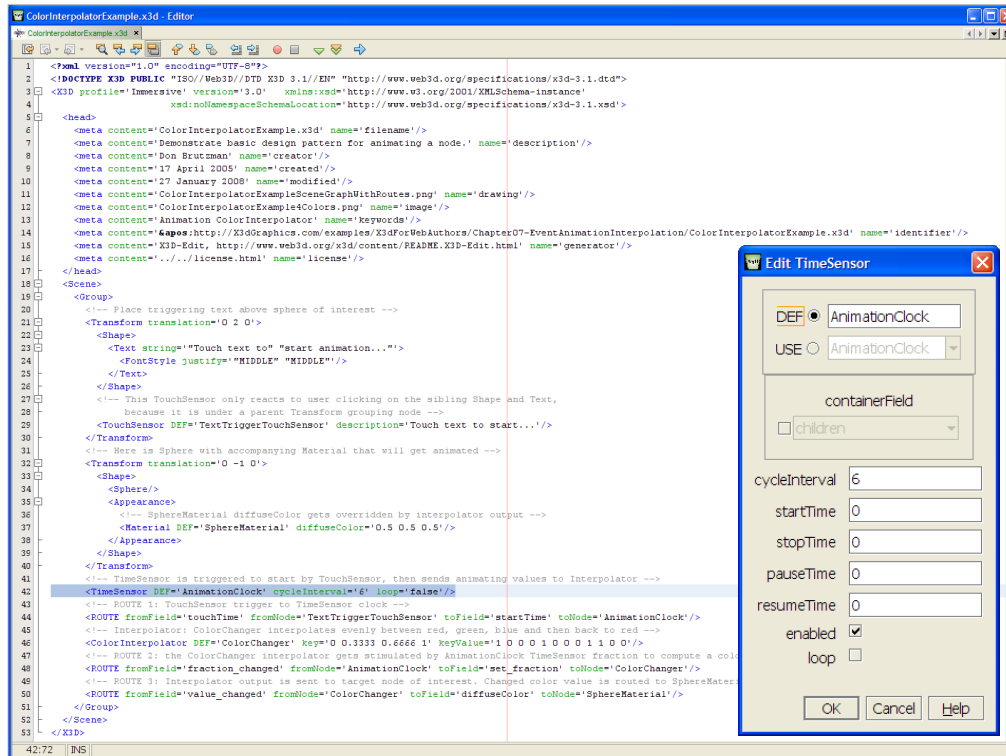



TimeSensor fields 1

- *enabled* controls whether node enabled or disabled
- *loop* is an SFBool indicating whether to continue looping indefinitely after first cycle is complete
- *cycleInterval* defines total loop duration in seconds, either for single-shot animation or looped repetition
- *startTime*, *stopTime* are provided (or contain) SFTIME values for when to start, stop respectively
 - ROUTE an SFTIME value to *set_startTime* or *set_stopTime*
- *cycleTime* field is sent an SFTIME output value upon completion of each loop

TimeSensor fields 2

- *isActive*, *isPaused* are output SFBool true/false events sent whenever the TimeSensor is set to run or paused
- *pauseTime*, *resumeTime* are SFTIME values for current clock time whenever paused or resumed
 - Corresponding boolean *isPaused* event is also sent, with value of true when paused and false when resuming
- *elapsedTime* output provides cumulative number of seconds since TimeSensor was activated and began running, without including paused time



| | |
|---|--|
|  TimeSensor | TimeSensor continuously generates events as time passes. Typical use: ROUTE thisTimeSensor.fraction_changed TO someInterpolator.set_fraction. Interchange profile hint: TimeSensor may be ignored if cycleInterval < 0.01 second. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| enabled | [enabled: accessType inputOutput, type SFBool (true/false) "true"] Enables/disables node operation. |
| cycleInterval | [cycleInterval: accessType inputOutput, type SFTIME CDATA "1.0"] cycleInterval is loop duration in seconds. Interchange profile hint: TimeSensor may be ignored if cycleInterval < 0.01 second. |
| loop | [loop: accessType inputOutput, type SFBool (true/false) "false"] Repeat indefinitely when loop=true, repeat only once when loop=false. |
| startTime | [startTime: accessType inputOutput, type SFTIME CDATA "0"] When time now >= startTime, isActive becomes true and TimeSensor becomes active. Absolute time: number of seconds since Jan 1, 1970, 00:00:00 GMT. Hint: usually receives a ROUTED time value. |
| stopTime | [stopTime: accessType inputOutput, type SFTIME CDATA "0"] When stopTime becomes <= time now, isActive becomes false and TimeSensor becomes inactive. Absolute time: number of seconds since Jan 1, 1970, 00:00:00 GMT. Hint: usually receives a ROUTED time value. |
| cycleTime | [cycleTime: accessType outputOnly, type SFTIME CDATA #FIXED ""] cycleTime sends a time outputOnly at startTime, and also at the beginning of each new cycle (useful for synchronization with other time-based objects). |
| isActive | [isActive: accessType outputOnly, type SFBool (true/false) #FIXED ""] isActive true/false events are sent when TimeSensor starts/stops running. |
| isPaused | [isPaused: accessType outputOnly, type SFBool (true/false) #FIXED ""] isPaused true/false events are sent when TimeSensor is paused/resumed. Warning: not supported in VRML97. |
| pauseTime | [pauseTime: accessType inputOutput, type SFTIME CDATA "0"] When time now >= pauseTime, isPaused becomes true and TimeSensor becomes paused. Absolute time: number of seconds since Jan 1, 1970, 00:00:00 GMT. Hint: usually receives a ROUTED time value. Warning: not supported in VRML97. |
| resumeTime | [resumeTime: accessType inputOutput, type SFTIME CDATA "0"] When resumeTime becomes <= time now, isPaused becomes false and TimeSensor becomes inactive. Absolute time: number of seconds since Jan 1, 1970, 00:00:00 GMT. Hint: usually receives a ROUTED time value. Warning: not supported in VRML97. |
| elapsedTime | [elapsedTime: accessType outputOnly, type SFTIME CDATA #FIXED ""] Current elapsed time since TimeSensor activated/running, cumulative in seconds, and not counting any paused time. Warning: not supported in VRML97. |

| | |
|------------------|---|
| fraction_changed | [fraction_changed: accessType outputOnly, type SFFloat CDATA #FIXED ""] fraction_changed continuously sends value in range [0,1] showing time progress in the current cycle. |
| time | [time: accessType outputOnly, type SFTIME CDATA #FIXED ""] Time continuously sends the absolute time (since January 1, 1970) for a given simulation tick. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

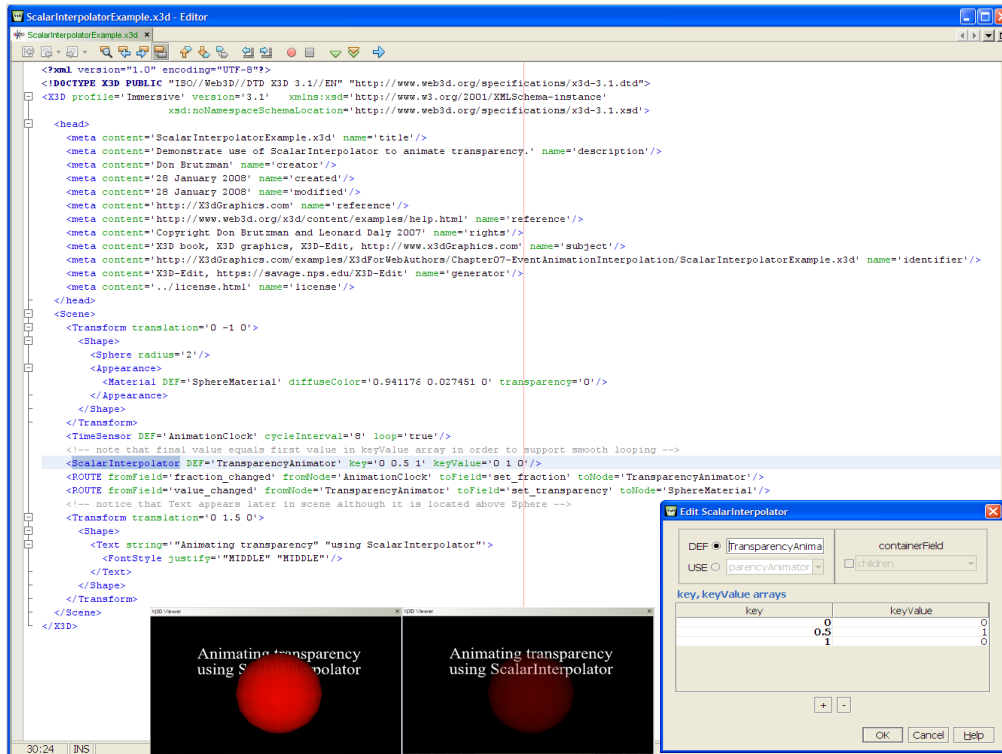
ScalarInterpolator node


Generates a scalar (single-valued) SFFloat for *value_changed* output

key and *keyValue* arrays contain SFFloat values

set_fraction determines input value to piece-wise linear function

- Percentage between bracketing *key*[*i*], *key*[*i*+1] values used to compute corresponding output *value_changed* as weighted average between *keyValue*[*i*], *keyValue*[*i*+1]
- Which is same algorithm for all interpolators



| | |
|---|---|
|  ScalarInterpolator | ScalarInterpolator generates piecewise-linear values that can be ROUTED to other Float attributes. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keys must match number of keyValues! |
| keyValue | [keyValue: accessType inputOutput, type MFFloat CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keys must match number of keyValues! |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type SFFloat CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

ColorInterpolator node

Generates a 3-tuple (triple-valued) SFColor for continuous *value_changed* output

key array contains SFFloat values

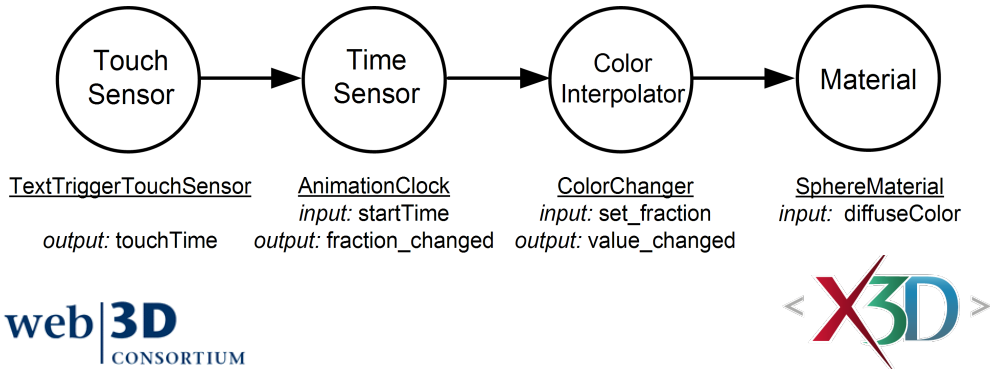
keyValue array contains SFColor values

Linear interpolation of red, green, blue (RGB) values is respectively performed for each bracketing *keyValue* pair

ColorInterpolator animation chain

Each node's output field matches data type of next node's input field

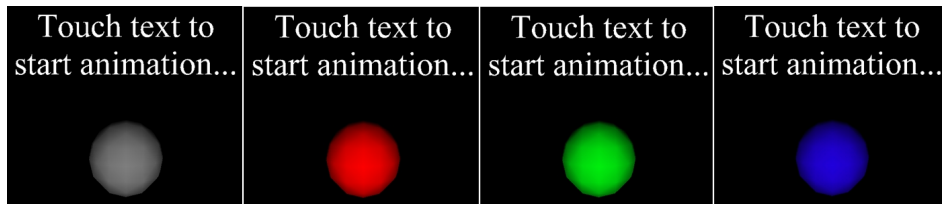
accessType outputOnly to inputOnly, initializeOnly also match



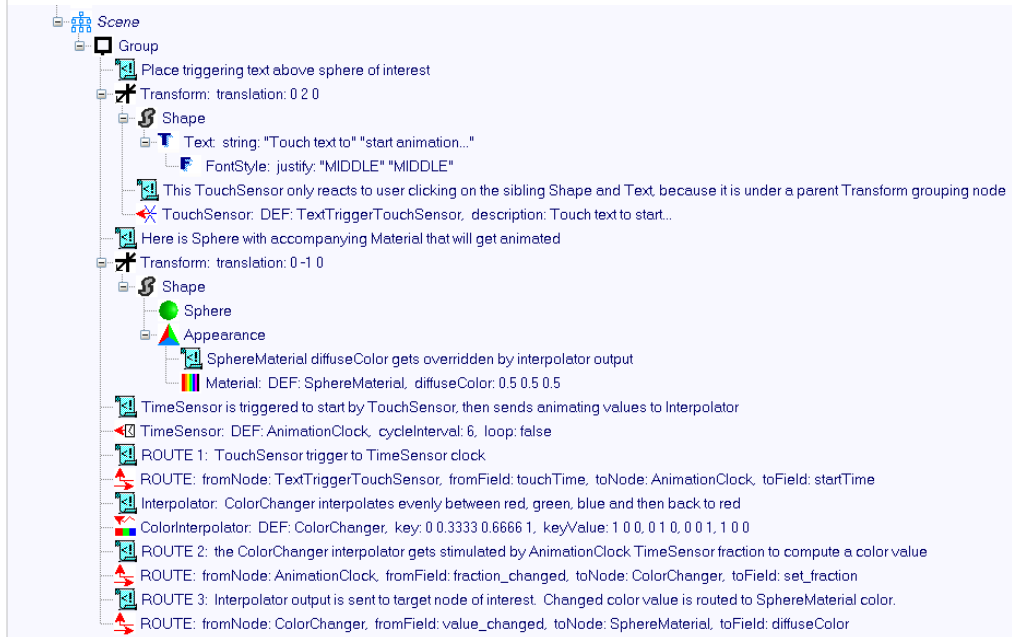
ColorInterpolator example output

Using the pointing device to select the text triggers the ColorInterpolator animation

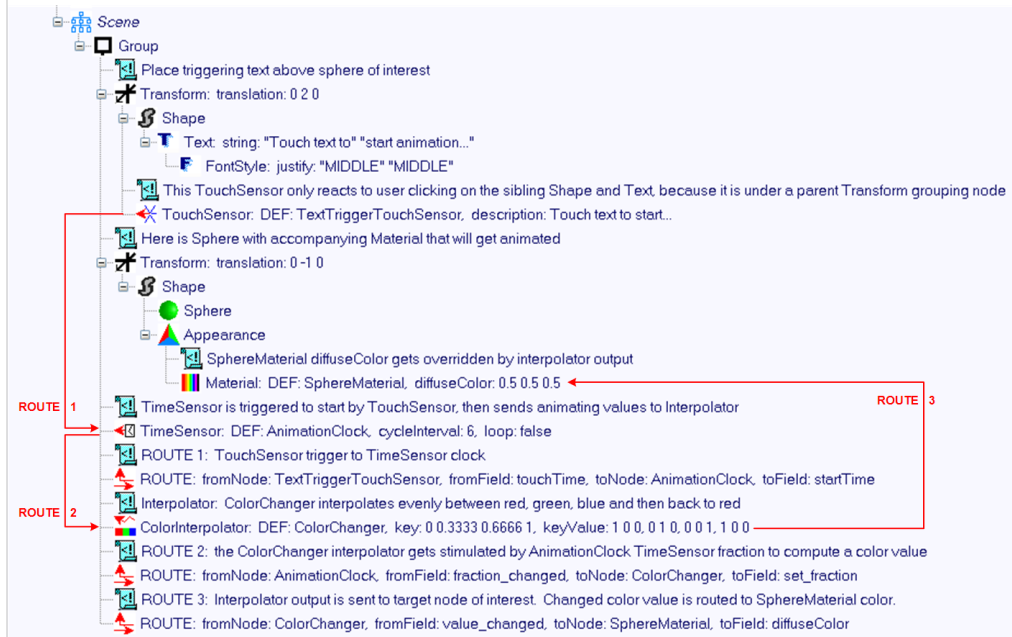
- Colors vary by linear interpolation of component red-green-blue RGB values

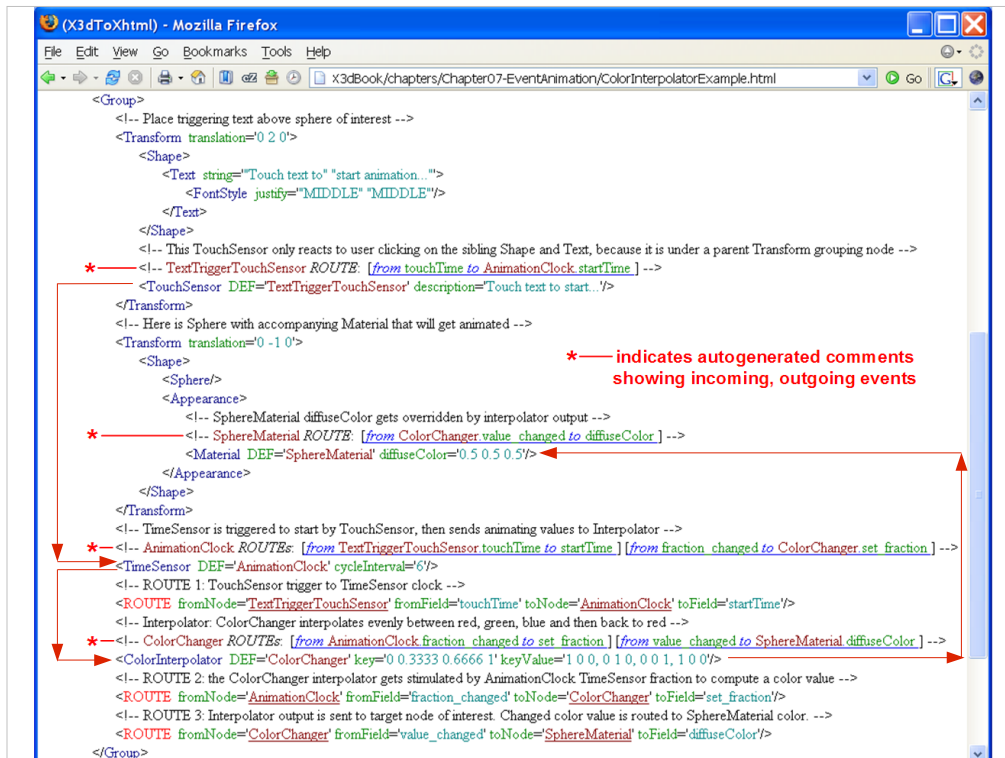


ColorInterpolator scene graph illustration

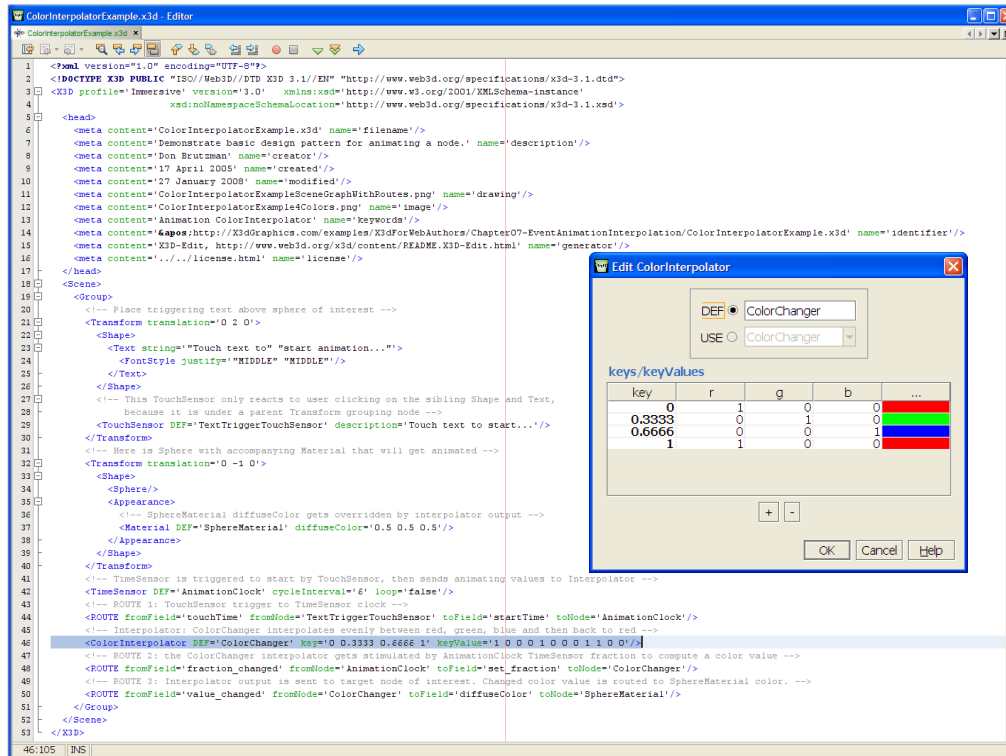



ColorInterpolator scene graph with ROUTEs





Pretty-printing a scene in HTML, printing it in landscape mode and then annotating it with ROUTE arrows is an excellent way to debug animation chains in a large scene.



| | |
|--|---|
| | |
|  ColorInterpolator | ColorInterpolator generates a range of Color values that can be ROUTEd to a <Color> node's color attribute. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction. Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keys must match number of keyValues! |
| keyValue | [keyValue: accessType inputOutput, type MFColor CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keys must match number of keyValues! |
| set_fraction | [set_fraction: accessType inputOnly, type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type SFColor CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |
| | |

OrientationInterpolator node

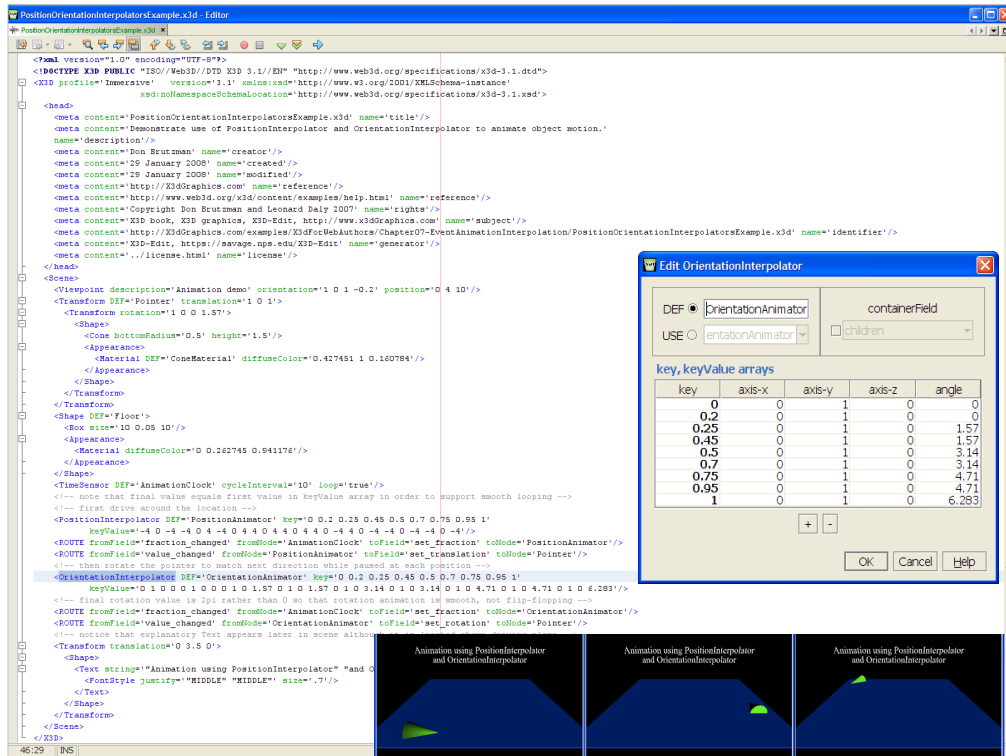
Generates a 4-tuple (four-valued orientation)
SFRotation for *value_changed* output


key array contains SFFloat values

keyValue array contains SFRotation values

- As always: same number of *key*, *keyValue* entries

OrientationInterpolator animates along shortest
path between the two normal vectors, also
computes linear average between two
corresponding angles, in *keyValue* array



| | |
|--|--|
| | |
|  OrientationInterpolator | OrientationInterpolator generates a series of rotation values Results can be ROUTed to a <Transform> node's 'rotation' attribute or another Rotations attribute Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keys must match number of keyValues! |
| keyValue | [keyValue: accessType inputOutput, type MFRotation CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keys must match number of keyValues! |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type SFRotation CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |
| | |

PositionInterpolator node

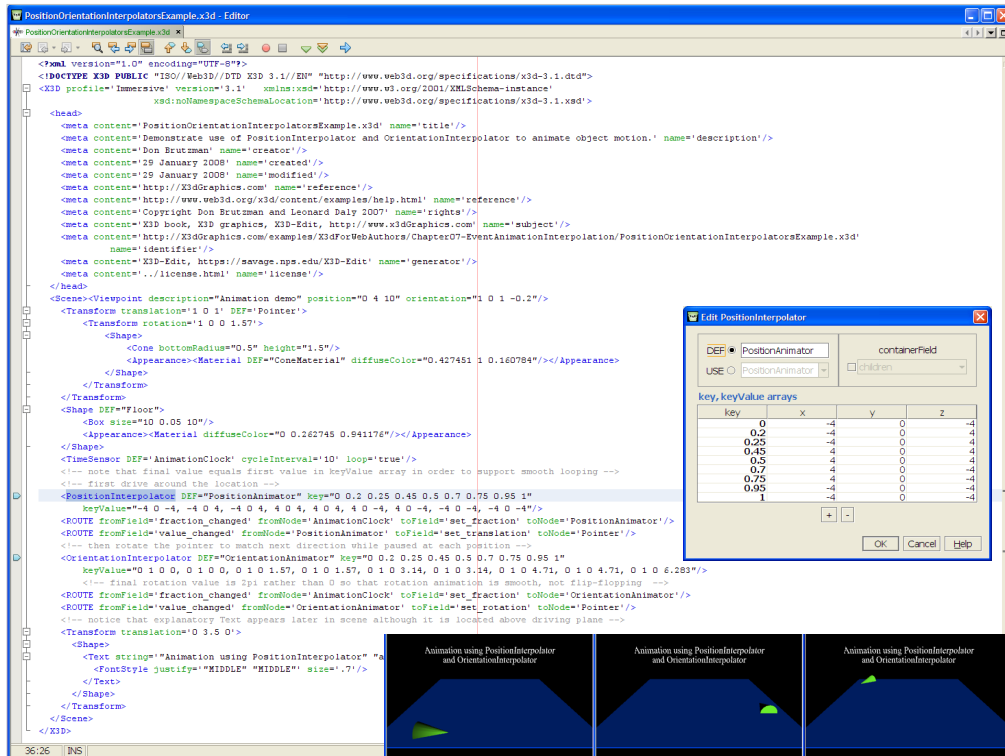
Generates a 3-tuple (three-valued floating point)
SFVec3f for *value_changed* output


key array contains SFFloat values

keyValue array contains SFVec3f values

- As always: same number of *key*, *keyValue* entries

PositionInterpolator computes weighted average
between corresponding x, y and z pairs in the
keyValue array



| | |
|---|---|
| | |
|  PositionInterpolator | PositionInterpolator generates a series of triplet values. Results can be ROUTED to a <Transform> node's 'translation' attribute or another Vector3Float attribute. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction. Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USING other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keys must match number of keyValues! |
| keyValue | [keyValue: accessType inputOutput, type MFVec3f CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keys must match number of keyValues! |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type SFVec3f CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |
| | |

PositionInterpolator2D node

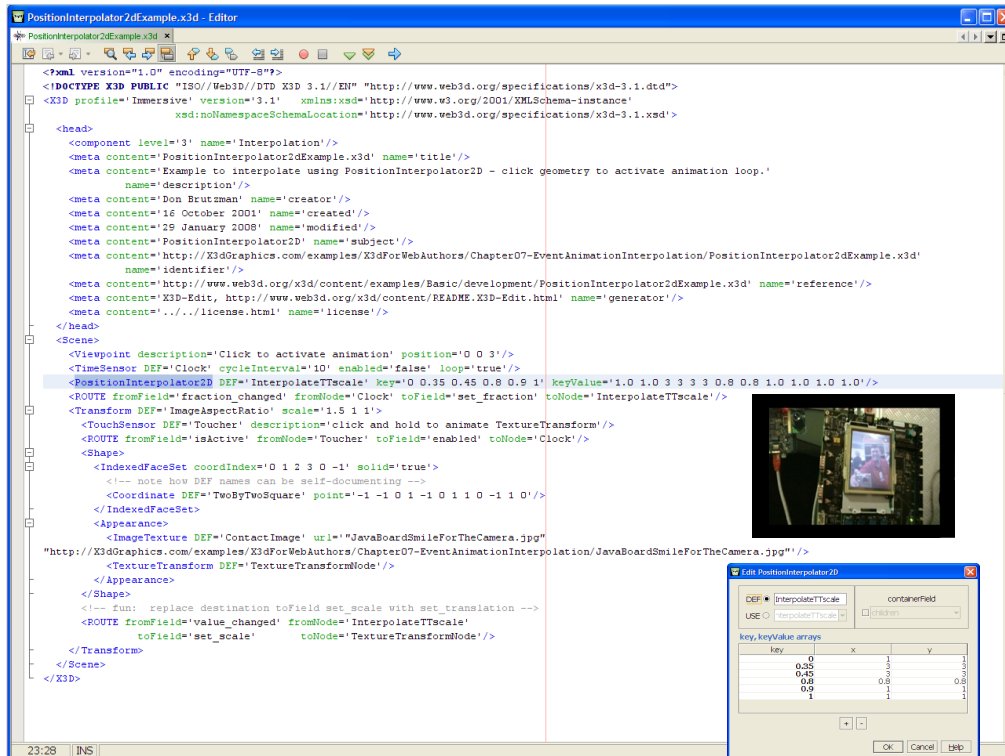
Generates a 2-tuple (two-valued floating point)
SFVec2f for *value_changed* output

key array contains SFFloat values

keyValue array contains SFVec2f values

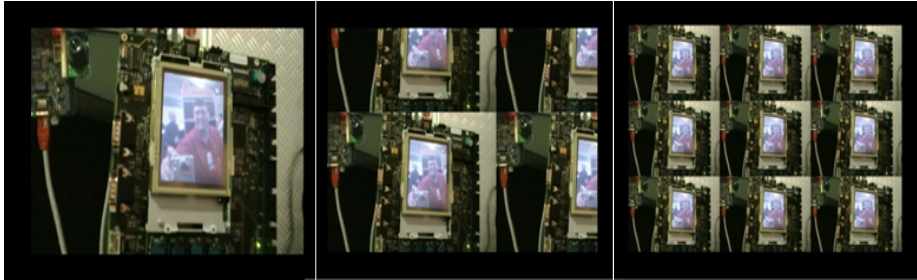
- As always: same number of *key*, *keyValue* entries

PositionInterpolator computes weighted average
between corresponding x and y pairs in the
keyValue array



<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter07-EventAnimationInterpolation/PositionInterpolator2dExample.x3d>

PositionInterpolator2D screen captures



Selecting the texture with the mouse pointer
starts the TextureTransform *scale* animation,
deselecting the texture stops the animation

web|3D
CONSORTIUM



<http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter07-EventAnimationInterpolation/PositionInterpolator/>

fun: replace destination toField *set_scale* with *set_translation* in the ROUTE

| | |
|---|--|
| | |
|  PositionInterpolator2D | PositionInterpolator2D generates a series of Vector2Float values that can be ROUTEd to a Vector2Float attribute. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction. Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue.key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| keyValue | [keyValue: accessType inputOutput, type MFVec2f CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue.key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type SFVec2f CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. Hint: keyValue.key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |
| | |

NormalInterpolator node

Generates a 3-tuple (three-valued floating point)
SFVec3f for *value_changed* output

key array contains SFFloat values


keyValue array contains SFVec3f values

- As always: same number of *key*, *keyValue* entries
- SFVec3f outputs: unit-normal vectors, magnitude=1

NormalInterpolator animates along shortest path
between the two normal vectors currently
being referenced in *keyValue* array

NormalInterpolator node X3D-Edit

[TODO: example needed]

| | |
|---|--|
| | |
|  NormalInterpolator | NormalInterpolator generates a series of normal (perpendicular) vector sets along the surface of a unit sphere ROUTE values to vector attribute of a <Normal> node or another Vector3FloatArray attribute Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keys must match number of keyValues! |
| keyValue | [keyValue: accessType inputOutput, type MFVec3f CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keys must match number of keyValues! |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type MFVec3f CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |
| | |

CoordinateInterpolator node

Generates n -tuple (multiple-valued floating point) array, MFFloat for *value_changed* output

key array contains n SFFloat values


keyValue array contains n MFFloat values

- As always: same number of *key*, *keyValue* entries
- Counting is very important for arrays of arrays!

CoordinateInterpolator computes weighted average between corresponding element pairs for each subarray in the *keyValue* array

CoordinateInterpolator node X3D-Edit

[TODO: example needed]

| | |
|---|--|
| | |
|  CoordinateInterpolator | CoordinateInterpolator generates a series of Coordinate values that can be ROUTED to a <Coordinate> node's 'point' attribute or another Vector3FloatArray attribute. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction. Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USEing other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| keyValue | [keyValue: accessType inputOutput, type MFVec3f CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type MFVec3f CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. Hint: keyValue/key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |
| | |

CoordinateInterpolator2D node

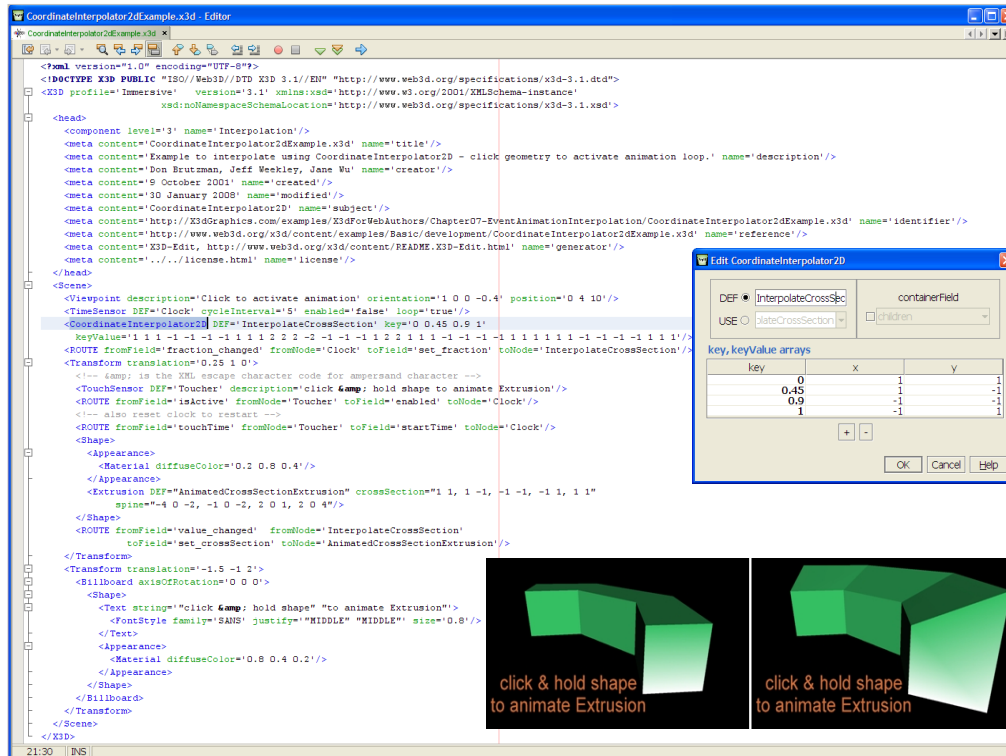
Generates 2-tuple (two-valued floating point) array, MFVec2f for *value_changed* output


key array contains SFFloat values

keyValue array contains MFVec2f values

- As always: same number of *key*, *keyValue* entries
- Counting is very important for arrays of arrays!

CoordinateInterpolator2D computes weighted average between corresponding x and y pairs for each subarray in the *keyValue* array



| | |
|--|--|
| | |
|  CoordinateInterpolator2D | CoordinateInterpolator2D generates a series of Vector2FloatArray values that can be ROUTEd to a Vector2FloatArray attribute. Typical input: ROUTE someTimeSensor.fraction_changed TO someInterpolator.set_fraction. Typical output: ROUTE someInterpolator.value_changed TO destinationNode.set_attribute. |
| DEF | [DEF ID #IMPLIED] DEF defines a unique ID name for this node, referencable by other nodes. Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED] USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children. Hint: USING other geometry (instead of duplicating nodes) can improve performance. Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| key | [key: accessType inputOutput, type MFFloat CDATA #IMPLIED] Definition parameters for linear-interpolation function time intervals, in increasing order and corresponding to keyValues. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| keyValue | [keyValue: accessType inputOutput, type MFVec3f CDATA #IMPLIED] Output values for linear interpolation, each corresponding to time-fraction keys. Hint: number of keyValues must be an integer multiple of the number of keys! Hint: keyValue key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| set_fraction | [set_fraction: inputOnly type SFFloat CDATA #FIXED ""] set_fraction selects input key for corresponding keyValue output. |
| value_changed | [value_changed: accessType outputOnly, type MFVec2f CDATA #FIXED ""] Linearly interpolated output value determined by current key time and corresponding keyValue pair. Hint: keyValue key integer multiple defines how many coordinates are sent in value_changed outputOnlys. |
| containerField | [containerField: NMTOKEN "children"] containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED] class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |
| | |

[back to Table of Contents](#)

Chapter Summary



Chapter Summary: Event Animation

ROUTE connections and animation

Animation as scene-graph modification

Event-animation design pattern: 10-step process

Interpolation nodes

- TimeSensor and event timing
- ScalarInterpolator and ColorInterpolator
- OrientationInterpolator, PositionInterpolator, PositionInterpolator2D and NormalInterpolator
- CoordinateInterpolator, CoordinateInterpolator2D



[back to Table of Contents](#)

References



References 1

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

- Chapter 7, Event Animation
- <http://x3dGraphics.com>
- <http://x3dgraphics.com/examples/X3dForWebAuthors>

X3D Examples Help

- <http://www.web3d.org/x3d/content/examples/help.html>



References 2

X3D Scene Authoring Hints

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

X3D Graphics Specification

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

Pocock, Lynn and Judson Rosebush,
The Computer Animator's Technical Handbook,
Morgan Kaufmann Publishers, 2001.



References 3

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

- <http://www.wiley.com/legacy/compbooks/vrml2sbk/cover/cover.htm>
- <http://www.web3d.org/x3d/content/examples/Vrml2.0Sourcebook>
- Chapter 08 – Animating Position Orientation Scale



Contact

Don Brutzman

brutzman@nps.edu

<http://web.nps.navy.mil/~brutzman>

Code USW/Br, Naval Postgraduate School
Monterey California 93943-5000 USA

1.831.656.2149 voice

1.831.656.7599 fax



Open-source license

Copyright (c) 1995-2008 held by the author(s). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the names of the Naval Postgraduate School (NPS) Modeling Virtual Environments and Simulation (MOVES) Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.