

**SONY®**

***Community Place Conductor 2.0  
Tutorial***

---

Copyright (c) 1998 Sony Corporation, All rights reserved  
Community Place is a trademark of Sony Corporation.

The use and reproduction of this software is permitted only within accordance with the terms of the Licence Agreement. The contents of this publication may be modified at any time without notice, and represent no affirmation to product specification or pursuit of obligations by Sony Corporation.

No part of this publication may be reproduced, transmitted, transcribed, stored, or translated to any other languages in any form or by any means, without prior written permission of Sony Corporation.

Windows, Windows NT are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries. Netscape Navigator is the trademark of Netscape Communications Corporation, USA. Other product and company names mentioned herein may be the trademarks or registered trademarks of their respective owners.

# Contents

<b>Preface .....</b>	<b>1</b>
<b>Creating Content .....</b>	<b>2</b>
<b>Turning a light ON/OFF by clicking a CONE. ....</b>	<b>6</b>
1.1 Creating a Geometry node (Cone) .....	6
1.2 Creating a Sensor node (TouchSensor) .....	7
1.3 Creating a Common node (PointLight) .....	9
1.4 Naming the node to be routed.....	10
1.5 Specifying a Route.....	11
1.6 Testing the operation .....	13
<b>Changing the color of a Cone when the mouse pointer moves onto the Cone. ....</b>	<b>14</b>
2.1 Creating a Geometry node (Cone) .....	14
2.2 Creating a Sensor node (TouchSensor) .....	14
2.3 Naming the node to be routed.....	15
2.4 Creating the Script node .....	15
2.5 Implement the program corresponding to the Script node (debug code only) .....	17
2.6 Specifying a Route to the Script node .....	18
2.7 Testing the Operation .....	18
2.8 Implement the program corresponding to the Script node (to specify a color) .....	19
2.9 Specifying a Route to the Material node .....	20
2.10 Testing the Operation .....	21
<b>Producing a Sound by Clicking Cone .....</b>	<b>23</b>
3.1 Creating a Geometry node (Cone) .....	23
3.2 Creating a Sensor node (TouchSensor) .....	23
3.3 Creating a Sound node.....	23
3.4 Testing the operation .....	24
3.5 Setting and routing the AudioClip node .....	27
3.6 Testing the operation .....	28

<b>Editing a Route in the Visual Edit Mode .....</b>	<b>29</b>
4.1 Creating a Geometry node (Cone) .....	29
4.2 Creating a Sensor node (TouchSensor) .....	29
4.3 Creating a Common node (PointLight) .....	29
4.4 Naming the node to be routed.....	29
4.5 Specifying a Route.....	31
4.6 Testing the operation .....	32
 <b>Keyframe Animation .....</b>	 <b>33</b>
5.1 Creating a Geometry node (Cone).....	33
5.2 Creating and Naming a TimeSensor Node .....	34
5.3 Animation Edit Mode .....	35
5.4 Move: Creating the First Keyframe.....	36
5.5 Move: Creating the Second Keyframe .....	38
5.6 Testing the Operation .....	39
5.7 Changing the Speed of the Animation.....	40
5.8 Rotate: Creating the First Keyframe.....	41
5.9 Rotate: Creating the Second Keyframe .....	43
5.10 Testing the Operation .....	44
 <b>PROTO Functions .....</b>	 <b>45</b>
6.1 Creating a table out of several primitives (Cylinder) .....	46
6.2 Creating PROTO .....	47
6.3 Reproducing a Table (Producing an Instance).....	48
6.4 Showing the internal field of PROTO to the Outside (1) .....	50
6.5 Showing the internal field of PROTO to the Outside (2) .....	53
6.6 Changing the Attributes of an Instance .....	53
 <b>Applicable Keyframe Animation .....</b>	 <b>55</b>
7.1 Creating a Geometry node (Cone) .....	55
7.2 Creating a TimeSensor Node .....	55
7.3 Animation Edit Mode .....	55
7.4 Varying Colors: Creating the First Keyframe .....	56
7.5 Varying Colors: Creating the Second Keyframe .....	59
7.6 Testing the Operation .....	60

7.7 Varying Shapes: Creating the First Keyframe.....	60
7.8 Varying Shapes: Creating the First Keyframe.....	63
7.9 Testing the Operation .....	64
7.10 Creating a Sensor node (TouchSensor) .....	64
7.11 Creating a Route .....	65
7.12 Testing the Operation .....	66
<b>References .....</b>	<b>67</b>

---

# **Preface**

---

This Tutorial shows you the procedure for creating simple content using Conductor 2.0. You may be able to follow the Tutorial without knowledge of VRML 2.0, but some knowledge will help you understand this document better.

# Creating Content

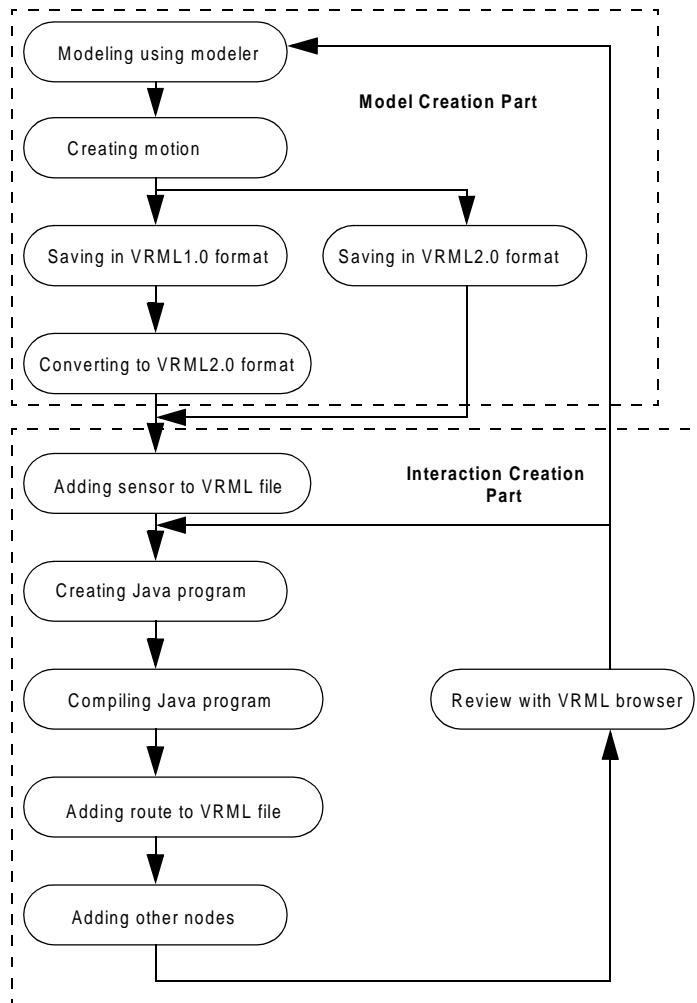


Figure 1: The general procedure for creating VRML2.0 content using Conductor

You can create content in VRML2.0 without Conductor by taking the following 2 major steps, as shown on Figure 1.

- **Step 1**

In step 1, create models. Models can be created using a conventional 3D modeler. If the 3D modeler you use enables you to program motions in the model as well (such as 3D Studio Max by AutoDesk), add them here. Then, convert the file into the VRML2.0 format to complete step 1.

- **Step 2**

In step 2, add interaction. Usually you will add the Sensor or Script nodes to the VRML file using a text editor. If the script is written in Java, it needs to be compiled. Add the Routes to the VRML file, and then use a VRML2.0 browser (such as the Sony Community Place Browser) to see how it works on the screen.



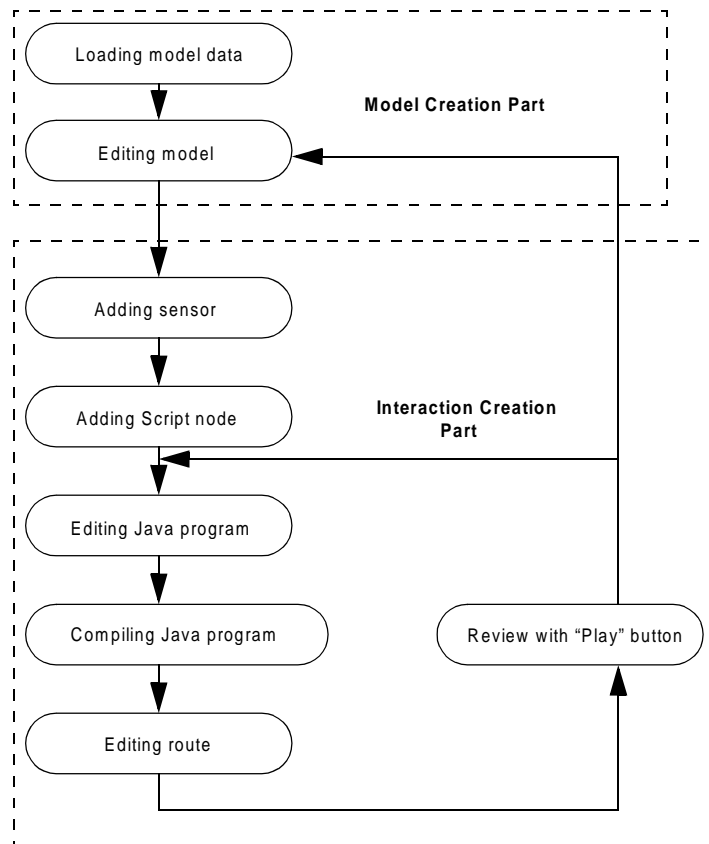



Figure 2: The general procedure for creating VRML2.0 content using Conductor

The chart shows you the procedure for creating content using Conductor. Basically, all the above steps can be achieved by a single tool, Conductor.

Conductor enables you to add sensors or edit route while seeing the results on the screen. The Script Expert function also allows you to create a template interactively (it will be explained later in this Tutorial). Conductor automatically runs the Java compiler, which frees you from manually compiling a Java file. To try out

your content, just press the Play button . The edit screen immediately changes to the browser screen where you can see the results.

Thus, Conductor greatly reduces the time required to create VRML2.0 content.

The following chapters will show you some examples of how to create simple content using Conductor.

**Note:** Grid lines are not shown in the 3D/2D View Windows in this Tutorial. To display/hide grid lines in Conductor, check the Environment setting dialog from Option/Environment menu.

## **Turning a light ON/OFF by clicking a CONE.**

The general procedure for attaining the above job is as follows:

- 1.1 Creating a Geometry node (Cone)
- 1.2 Creating a Sensor node (TouchSensor)
- 1.3 Creating a Common node (PointLight)
- 1.4 Naming the node to be routed
- 1.5 Specifying a Route
- 1.6 Testing the operation

The following sections describe each of these steps.

### **1.1 Creating a Geometry node (Cone)**

Let's start with placing objects within the world for the user to see and interact with.

Start Conductor, open an empty world by choosing File/New, and then create a Cone. To create a Cone, first select Cone from the Geometry tab inside the Community Place Conductor window (the Main window in the following pages). Next, click on the 3D Perspective View (3D View in the following pages) window. A Cone is added to the 3D View Window, just in front of your eyes as shown in Figure 3 .

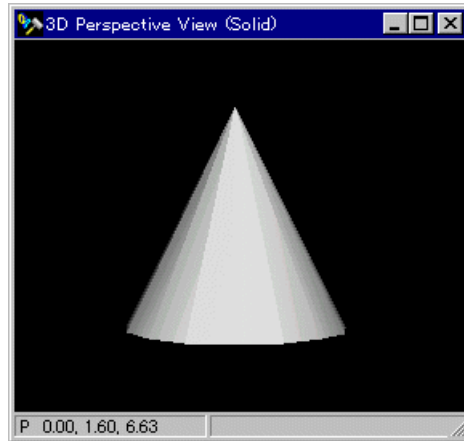


Figure 3: Creating a Cone

Note: You are allowed to create only the objects provided by the VRML2.0 basic nodes or the library window. Other objects created with the other modeling package can also be created by converting them into the VRML2.0 format.

## 1.2 Creating a Sensor node (TouchSensor)

To add motion to an object or to have it interact with the user, you have to create a Sensor node. The Sensor node is used to detect a preprogrammed condition, and notifies it to other nodes as an event. For this step, create a TouchSensor to detect the mouse operation. By placing a Touch Sensor to the Transform above the object, you can detect a mouse activity on that object.

To add a TouchSensor to a selected node, select the node and then create a Touch-Sensor. First select Transform at the hierarchical position above the Cone to select it in the Scene Graph Window. Next click the Sensor tab in the Main window, and choose TouchSensor. Then click on the 3D View window.

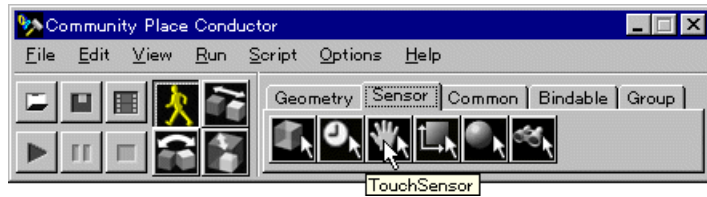


Figure 4: Choosing TouchSensor

A TouchSensor has been added to the Transform, as shown in the Scene Graph Window (Figure 5 ). You can check to see in the Scene Graph Window that a TouchSensor has been added to the Transform.

TouchSensor will be applied on all objects below that Transform. In this example, if a Box exists under the same Transform as Cone, TouchSensor will activate whenever Cone or Box are clicked.



Figure 5: Confirming TouchSensor from within the Scene Graph window

**Note:** A TouchSensor has been added to a simple object, the Cone, in this example, but you can also add TouchSensors to more intricate objects.

### 1.3 Creating a Common node (PointLight)

Within the operational flow described in VRML2.0, the Sensor node notifies another node of the event it detects, then the node that has received the event reacts appropriately.

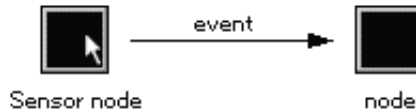


Figure 6: Process flow

Create a Light as the node to receive events. You will find Light on the Common tab. Select the 3D View window, and press the Down-arrow key to move your eye position further from the object. Creating a PointLight at this position should illuminate the Cone as shown on Figure 7 .

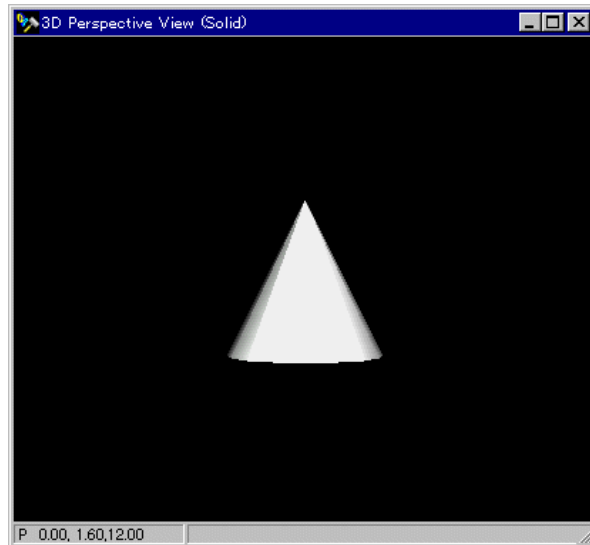


Figure 7: PointLight is placed (Cone is illuminated)

## 1.4 Naming the node to be routed

Next, you should route from the TouchSensor to the PointLight. To transfer the event from the Sensor node to the other nodes, they have to be connected. This connection is called a route. Before routing nodes, you need to give the DEF names to the nodes.

You can name them from within the Attribute Window. The Attribute Window is used to change the currently selected object's attributes. To name the nodes, select TouchSensor in the Scene Graph Window, and enter DEF names in the Attribute Window. Name the TouchSensor "TS1" in this example, as shown on Figure 8 . Type "TS1" in the DEF name box and press the Return key.

Name the PointLight node "PL1" in the same way by selecting PointLight in the Scene Graph Window. Set the default status of the Light to OFF. You can do this by double-clicking the PointLight's ON field in the Attribute Window. (It will initially be FALSE.)

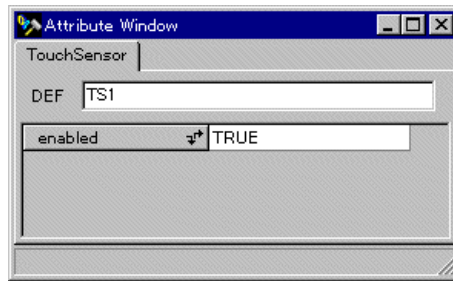


Figure 8: TouchSensor has been named

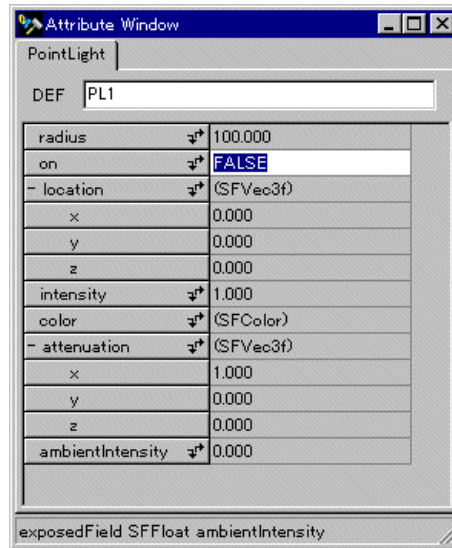


Figure 9: PointLight has been named and turned off

## 1.5 Specifying a Route

Now you are ready to route the nodes. Let's specify a route from the TouchSensor node (TS1) to the PointLight (PL1), so that an event detected by TouchSensor will be passed to the PointLight.

To route the two nodes with Conductor, display the Route Window. Click View/Route in the Main window to display the Route Window. The Route Window provides two edit modes: Dialog and Visual. Here, you will be shown the procedure for using the Dialog edit mode. For information about Visual edit mode, see Chapter 4. It's much easier to edit with the Visual mode than with the Dialog mode. But using the Dialog edit mode, you will better understand the routing mechanism.



First select the Dialog edit mode by clicking the left-most button located at the bottom of the Route Window. In Dialog edit mode, the Route Window will appear as shown on Figure 10 . The eventOut name is specified in the upper box, and the eventIn name in the lower box. Let's click "TouchSensor" in this example. Specify "TS1" as the eventOut node name and select "isActive" for eventOut. The event type "SFBool" appears on its right side, indicating that the type of event passed by "isActive" will be SFBool. Next, specify "PL1" as the eventIn node name and select "on" for eventIn.

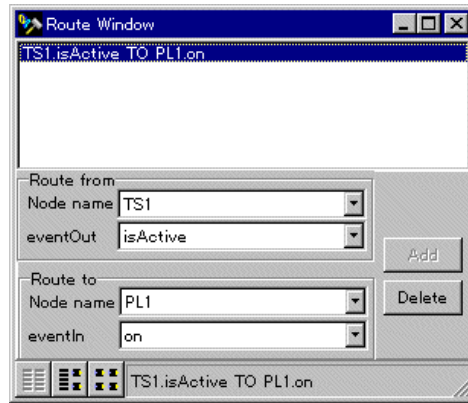



Figure 10: Routing from TouchSensor(TS1) to PointLight(PL1)


If you have selected all the necessary items, press the Add button. The Route will be established.

## 1.6 Testing the operation

Conductor allows you to emulate the browser just by pressing the Play button. Thus, you can check to see how the world works in the edit screen, without starting the VRML browser. You can also go back immediately to the normal edit mode by pressing the Stop button.

Test what you have done. Press the Play button 

In the above example, by clicking the cone, the TouchSensor node attached to it detects the click and generates an event. This event is passed to the Light node as specified in the route. The Light node receives the event, changes the condition detected by TouchSensor, and then turns the light on. Releasing the mouse button, the event is passed in the same way, and the light will be turned off.

You can restore the normal edit mode by pressing the Stop button 

To save the file you are working on, specify its filename with Save As on the File menu. You can also browse the file with Community Place Browser.

A simple behavior, such as the one shown in the above example, can easily be added by making effective use of node and routing without programming.

The following chapters will show you the procedure for more complicated jobs.

## **Changing the color of a Cone when the mouse pointer moves onto the Cone.**

In the previous chapter, the event generated by Sensor was directly passed to the other object. VRML2.0, however, enables more intricate operations to be achieved by adding a Script node between the nodes.

Let's think about changing the color of a Cone when the mouse pointer comes onto the Cone, for example. In VRML2.0, colors of objects can be specified with the Material attribute. Material has more than one field. You can change the color of an object by setting the diffuseColor field. In this example, we want the diffuseColor to be set by a user-induced event. Let's use the Script node to set the diffuseColor here.

Steps 2.1 and 2.2 are the same as those in the previous chapter, and will be skipped here. Restart Conductor, and create a Cone and TouchSensor in preparation. Name the TouchSensor "TS1" as well.

- 2.1 Creating a Geometry node (Cone)
- 2.2 Creating a Sensor node (TouchSensor)
- 2.3 Naming the node to be routed
- 2.4 Creating a Script node
- 2.5 Describing the program corresponding to the Script node (debug code only)
- 2.6 Specifying a Route for the Script node
- 2.7 Testing the operation
- 2.8 Describing the program corresponding to the Script node (to specify a color)
- 2.9 Specifying a Route to the Material node
- 2.10 Testing the operation

Each step is explained in the following sections.

## 2.3 Naming the node to be routed

Before routing, you were requested to name the nodes. In the previous chapter, you named TouchSensor and PointLight. In this chapter, you are requested to name the Material.

The Material node is contained in the Appearance node under VRML 2.0. Conductor also includes information about the object's material properties in the Appearance Window. The Appearance Window can be displayed by clicking View/Appearance on the menu in the Main window. The Appearance Window is used to edit the appearance (material, texture, and so on) of the currently selected object. Let's name the material of the Cone here. First select Cone, then type CONE1\_MAT as the name of the material by using the material tab in the Appearance Window.

**Note:** To set DEF name, type the name into the DEF name edit area and press Enter key.

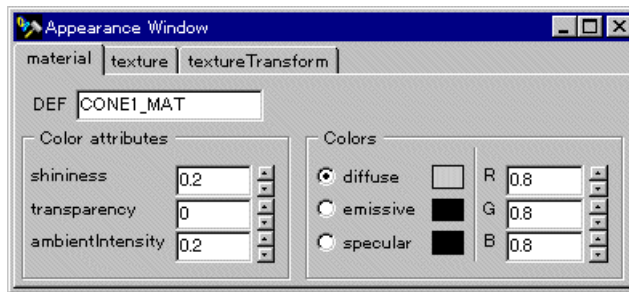


Figure 11: Naming material

## 2.4 Creating the Script node

VRML2.0 is provided with the Script node that is used to establish the interface with the external script program. More intricate behaviors can be described by combining the Script node and the external script programs (Conductor uses Java). The Script node, however, is a little difficult to handle, because it is closely connected with the external script programs and more than one input/output can be specified on it.

**Note:** You must have JDK 1.1.x installed to use Script.

You don't have to worry about these details. Conductor provides you with the Script Expert function that helps you easily create the Script node. Specifying input/output of Script, the Script Expert automatically outputs the template of the script program. You can also edit the input/output fields from within the Attribute Window.

Start creating the Script node. First choose Script/Expert in the Main window. The dialog box to create a Script appears on the screen. DEF at the top indicates the name of Script node. In order to route to the Script node, it must be named. Let's name the Script "SC1." Class Name below DEF indicates the class name for Java. In this example, specify "SC1" for Class Name. Next, define event input/output of the Script node. You are to input events from TouchSensor. Select SFBool as the eventIn Data Type (an event generated by TouchSensor will be passed in the SFBool data type), and specify "inBool" as the field name. You may want to control the color of Material by using eventOut. Select SFColor, and specify "outColor" as the field name. All the necessary settings for the Script node have been completed as shown on Figure 12 .

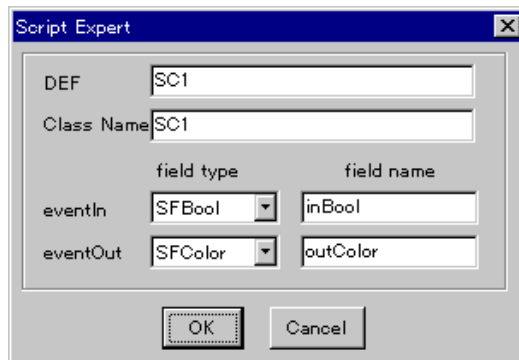


Figure 12: Creating the Script node with Script Expert

Press the OK button. The Script node and the program template corresponding to the Script node will be created.

## 2.5 Implement the program corresponding to the Script node (debug code only)

You are now ready to edit the newly defined Script node. The template of it, however, is only defined in its basic input/output aspects. You can promptly start writing the program to change the object color, but it is better to test if the program works properly by using the debug code beforehand.

Take a close look at the edit screen. The field name specified for eventOut is now defined as a private variable, *private SFColor m\_outColor*. The code used to initialize the variable in the initialize() method is also included. The method, \_inBoolCB(), is called when an event occurs on eventIn. The programs initiated by input from the Sensor must be described within the \_inBoolCB() method. Let's describe the code used to display input values. Add the following debug code to \_inBoolCB().

```
System.out.println("_inBoolCB() called: " + ev.getValue());
```

With this code, the input values will be displayed. After adding the code, make sure to compile it. Choose Compile from the pop-up menu by clicking the right mouse button. The compile status messages will appear in the Message Window. If the compilation terminates successfully, "Done" appears. If a compile error message appears, correct the error and compile again.



Figure 13: Adding the debug code to Script

## 2.6 Specifying a Route to the Script node

Routing is also necessary to activate Script. Let's route the TouchSensor node to the Script node here.

When the mouse pointer comes over the Cone, this event has to be passed to the Script node to induce the color change. The event that occurs when the mouse pointer comes onto the object (that TouchSensor manages) is "isOver." Specify "inBool" in the field of the Script node to receive the event, as specified previously with the Script Expert function. After the route is added, the Route Window will appear as follows:



Figure 14: Routing from TouchSensor to Script

## 2.7 Testing the Operation

Press Play to test the operation. If debug code has been included, the following message should appear if the above procedure is working properly.

To display the output messages written in Java, display the Java console window by using the View/Java Console menu in the Main window. Then press the Play button. Notice that when you move the mouse pointer onto the Cone, "true" will appear; if the mouse moves away from the Cone, "false" will appear.

In the above example, if the mouse pointer comes onto the Cone, the TouchSensor node attached to the Cone will detect it and cause an event. The event caused by TouchSensor will be passed to the Script node as specified in the route. The Script node receives the event, then performs the method. Since *System.out.println()* within the method is performed, the results are displayed in the java console output window.



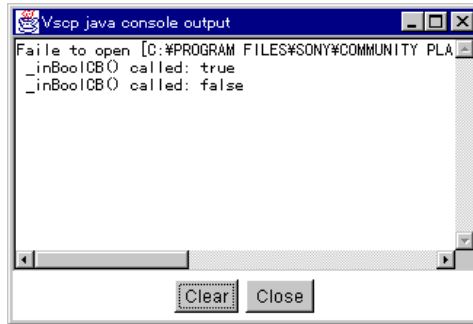


Figure 15: Displaying debug messages in the java console output window

**Note:** The above messages will be displayed when debugging the Script program.

## 2.8 Implement the program corresponding to the Script node (to specify a color)

You have confirmed that the event is properly passed to the Script node. Next, implement the code to change the color. Add the following code to `_inBoolCB()`:

```
float r[] = {1.0f, 0.0f, 0.0f};
float g[] = {0.0f, 1.0f, 0.0f};
if (ev.getValue()) m_outColor.setValue(r);
else m_outColor.setValue(g);
```

This will cause the Script node to output red if the mouse pointer comes onto the Cone (`ev.getValue()` will be true), or green if it moves away from the Cone (`ev.getValue()` will be false). By writing the value to the field on `eventOut` (in this case, `m_outColor`) and routing it to `Material`, you can change the color of the Cone.

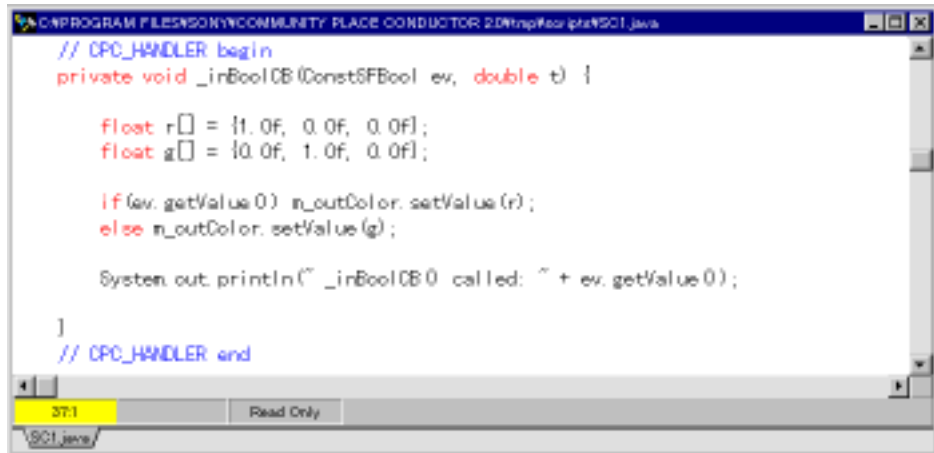


Figure 16: The code to change the color has been programmed.

## 2.9 Specifying a Route to the Material node

To actually change the color of the Cone, you have to add a route from the Script node to the Material node. Route SC1 outColor to CONE1\_MAT diffuseColor in the Route Window as follows:

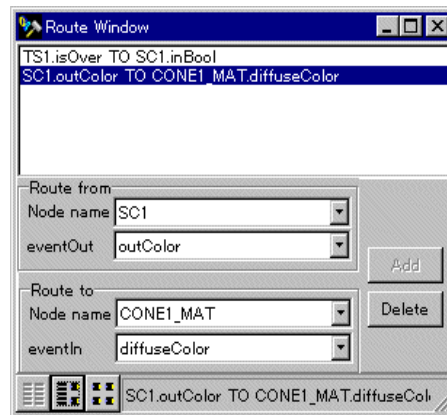


Figure 17: Routing from the Script node to the Material node

This, along with the routing you made in the previous section, will generate the event to be passed from the TouchSensor node (TS1) to the Script node (SC1), and then from the Script node to the Material node (CONE1\_MAT).

## 2.10 Testing the Operation

To test what you have done press the Play button

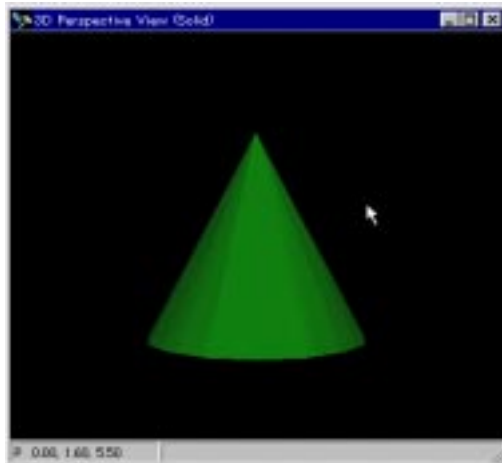
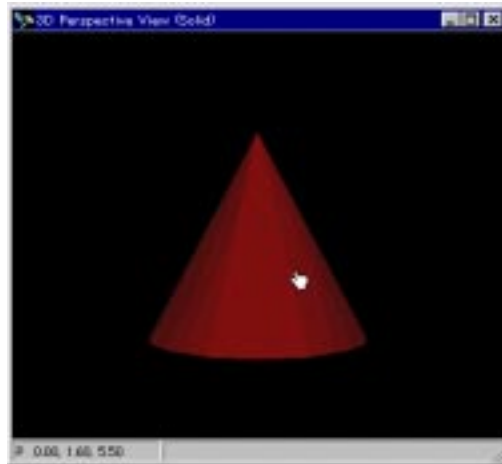


Figure 18: Cone turns red if the mouse pointer comes on it (top) and turns green if it moves away (bottom)

In the above example, by clicking the Cone, the TouchSensor node attached to the Cone will detect the mouse click and generate an event. The event generated by the TouchSensor will be passed to the Script node as specified in the route. The Script node receives the event, then performs the method. The method sets red in the output field if the Script node receives the input event (true). The color set in the output field then be set to the Material node of the cone, which causes the cone to turn red. In the same way, if the mouse pointer moves away from the Cone, TouchSensor generates an event (false). The Script node receives the event, and the method sets green in the output field, causing the Cone to turn red as specified in the route.

## **Producing a Sound by Clicking Cone**

You can produce a sound by using the Sound node under VRML2.0. Conductor also enables you to add the sound node to a world by dragging-and-dropping items from the sound library.

The general procedure for attaining the above job is given below. First confirm the default operation of the Sound node. Next edit attributes or specify the route. Then test the operation. Steps 3.1 and 3.2 are the same as those in the previous chapters, and will be skipped here. In the mean time, restart Conductor, and create a Cone and a TouchSensor. Name TouchSensor "TS1."

- 3.1 Creating a Geometry node (Cone)
- 3.2 Creating a Sensor node (TouchSensor)
- 3.3 Creating a Sound node
- 3.4 Testing the operation
- 3.5 Setting and routing the AudioClip node
- 3.6 Testing the operation

Each step is explained in the following sections.

### **3.3 Creating a Sound node**

Before producing a sound, a Sound node needs to be created. Conductor enables you to add a Sound node by dragging and dropping from the Sound library (the Sound tab is located at the top of the Resource Library Window) to the 3D View Window.

If you have not already done so, create a Cone and TouchSensor. You have to make sure that nothing is currently selected. (If any object is currently selected, choose Unselect from the pop-up menu within the Scene Graph Window.)

Check to see if you can see the Cone in the 3D View window. Select Classic from the Sound library within the Resource Library Window. Then, drag-and-drop it onto the 3D View window. The Sound node has been added to the world. You can confirm the addition by checking the Scene Graph Window.

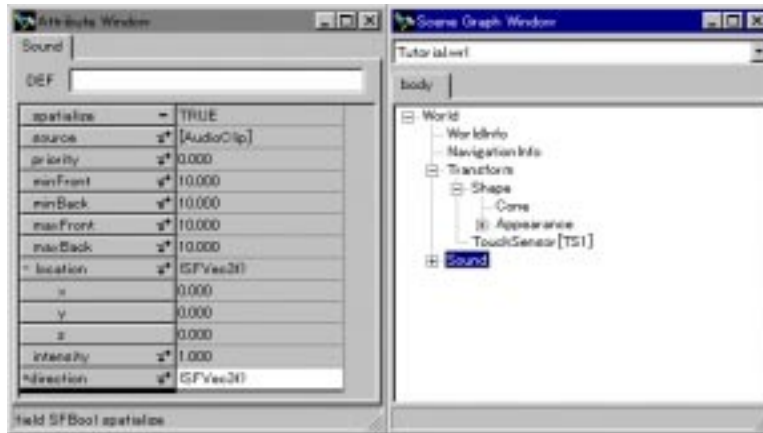


Figure 19: Dragging-and-dropping Sound from the library to the world

The Sound node allows you to specify the sound area by entering the values in the maxBack and maxFront fields. They are both set to 10 by default. You can hear the sound only within the area.

### 3.4 Testing the operation

You can hear the sound by pressing the Play button. As soon as you drag and drop a Sound, you will be inside the Sound area where a sound can be produced by pressing the Play button.

Press the Play button. Can you hear the classic sound? By walking through the 3D window with a mouse, the volume and the balance (between the right and left) will vary depending on where you are. This is because the spatialize field of the Sound node is set to TRUE by default.

Return to the edit mode by pressing the Stop button to see the sound area. To do this, choose Sound in the Scene Graph Window, and the bounding box indicating the sound area starts blinking. To obtain the whole view, zoom out the view.

To see from the top-down view, choose Parallel View/Top View from the pop-up menu in the 3D View window. To obtain the whole view, move our eye position/direction. (By pressing Shift and the left mouse button at the same time, you can zoom out by moving the mouse towards the lower portion of the screen.) The following pictures show you the sound area viewed from the different angles.

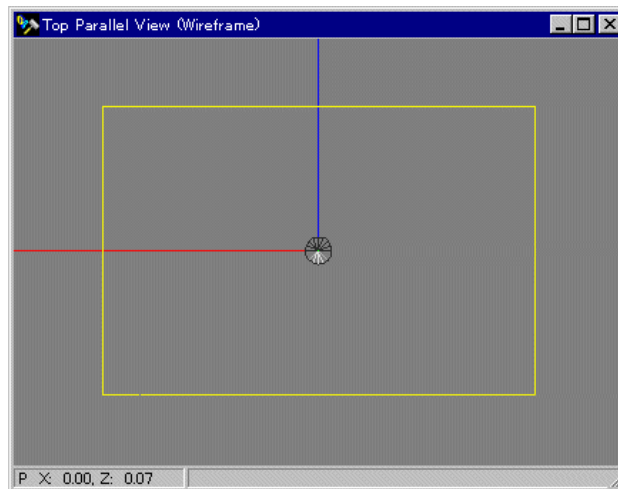
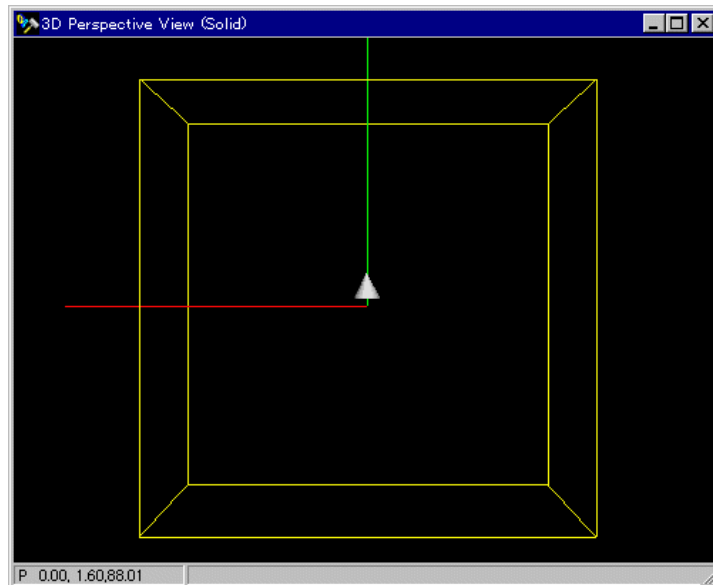


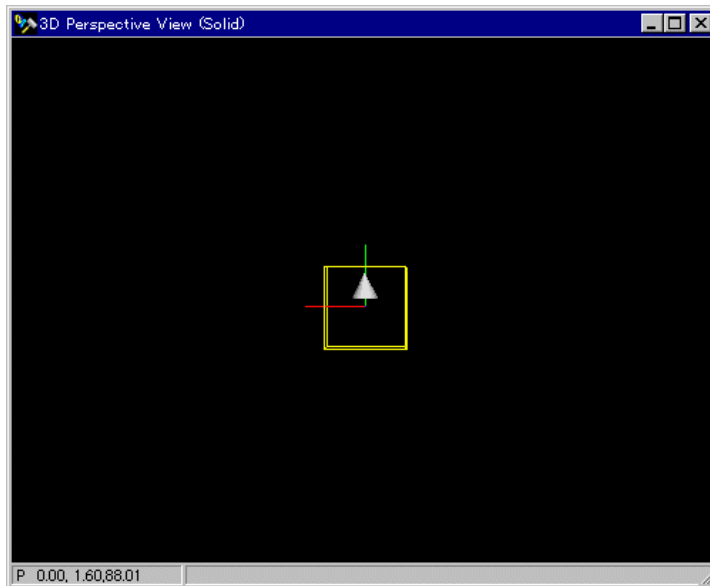
Figure 20: Sound area viewed from different angles

Note: The actual sound area is oval. However, it is represented by a cube on the screen.



If you walk through the 3D View window after pressing the Play button, you will hear the sound when coming inside the sound area, and will not hear the sound when coming out of the area.

Note: (TIP) You can specify the area by entering values in the Attribute Window. Conductor also allows you to move, rotate, or scale the area by changing the mouse mode in the Main window. (With the mouse, you are only allowed to scale it equally in all the directions.)



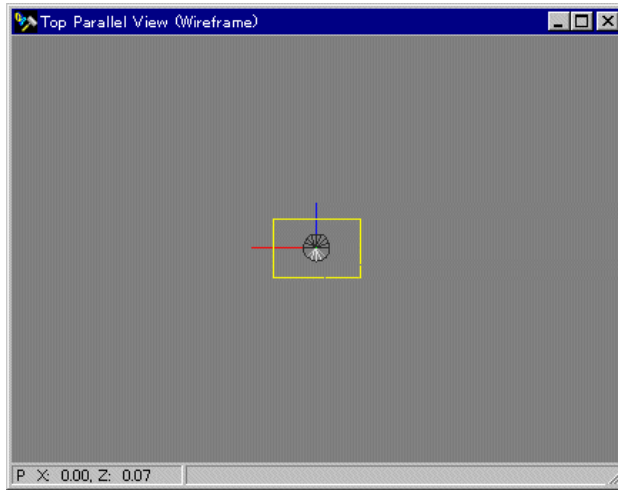


Figure 21: Reducing the size of the area with the mouse (This operation is not required.)

### 3.5 Setting and routing the AudioClip node

After dropping a Sound onto the world, you could hear a sound simply by moving inside the sound area. This is closely related to the settings of the AudioClip node attached to the Sound node: `startTime=0`, `stopTime=0`, `loop=TRUE`. Under VRML2.0, `startTime` indicates the start time and `stopTime` indicates the stop time. If `startTime`  $\geq$  `stopTime`, `stopTime` will be invalid. If `loop=TRUE` as well, the sound will be produced forever. Now, you have to disable the sound to be produced until the Cone is clicked by specifying `startTime=-1`

To enable the sound to be produced if the Cone is clicked, set *startTime* to the time the Cone is clicked. Name the AudioClip node "AC1," and then route from *touchTime* of the TouchSensor node (TS1) to *startTime* of the audioClip node.

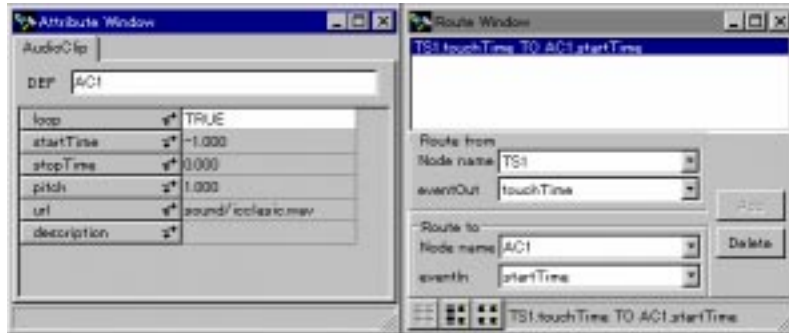


Figure 22: Changing the attribute for AudioClip and routing from TouchSensor to AudioClip

### 3.6 Testing the operation

Let's test what you have done. Press the Play button and click the Cone. You should hear the sound.

In the above example, the TouchSensor node attached to the Cone detect the mouse click and generates an event. The event passes the time the Cone was clicked (relative time from 00:00:00 GMT Jan 1 1970) to startTime of the AudioClip node. If startTime is set, the sound begins. The sound will be produced endlessly because loop of AudioClip is set to TRUE.

You cannot simply stop the sound here. To stop the sound, you have to write the program by using the Script node. This is an assignment for you. Please try.

**Note:** If you move out of the sound area, you will no longer hear the sound.

## **Editing a Route in the Visual Edit Mode**

The Route Window provides two edit modes: Dialog and Visual. You have learned how to edit in the Dialog mode in Chapter 1 through Chapter 3. In this chapter, however, you will learn how to perform the same job in the Visual edit mode. But using the Dialog edit mode, you will better understand the routing mechanism. It is recommended that you first try the Dialog edit mode, then switch over to the Visual edit mode. In the Visual edit mode, you can visually check to see on the screen how nodes are linked to one another, which will help you edit routes much more easily when there are a large number of nodes.

This chapter shows you the same procedure as that explained in Chapter 1, but in the Visual edit mode. The general procedure for attaining the above job will be as follows:

- 4.1 Creating a Geometry node (Cone)
- 4.2 Creating a Sensor node (TouchSensor)
- 4.3 Creating a Common node (PointLight)
- 4.4 Naming the node to be routed
- 4.5 Specifying a Route
- 4.6 Testing the operation

Steps 4.1 through 4.3 are the same as steps 1.1 through 1.3 in Chapter 1, and will be skipped here. Restart Conductor, and create a Cone and TouchSensor in preparation. Name the TouchSensor "TS1."

### **4.4 Naming the node to be routed**

The rest of this chapter differs from Chapter 1.

In the Dialog edit mode, you have to name a node in the Attribute Window in order to create its route under that name in the Route Window. In the Visual edit mode, however, you can edit routes visually and intuitively with drag and drop, in the Route Window.

First set the Route Window to the Visual edit mode. You can switch the edit modes by clicking one of the 3 buttons located at the bottom of the Route Window. Let's click the button at the right-most position here, which displays objects as icons. Drag the PointLight in the Scene Graph Window to the Route Window.

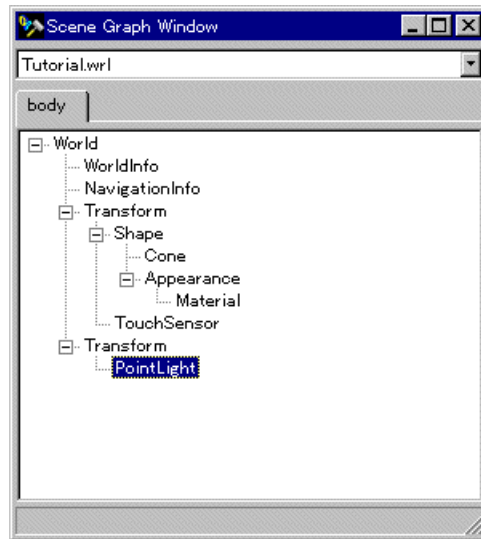


Figure 23: PointLight in the Scene Graph Window. Drag it to the Route Window.

The PointLight icon is created and automatically named "PointLight\_00" in the Route Window. You can rename it if you like.

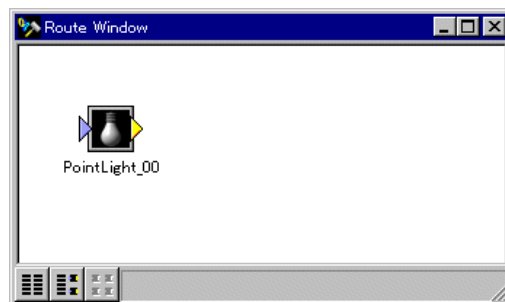


Figure 24: PointLight Automatically Named

## 4.5 Specifying a Route

Now you are ready to route the nodes. Drag a node that generates an event (such as TouchSensor) onto a node (such as PointLight) to receive the event in the Route Window.

Let's drag the TouchSensor in the Scene Graph Window to the Route Window here, and drop it onto the PointLight icon created in 4.4.

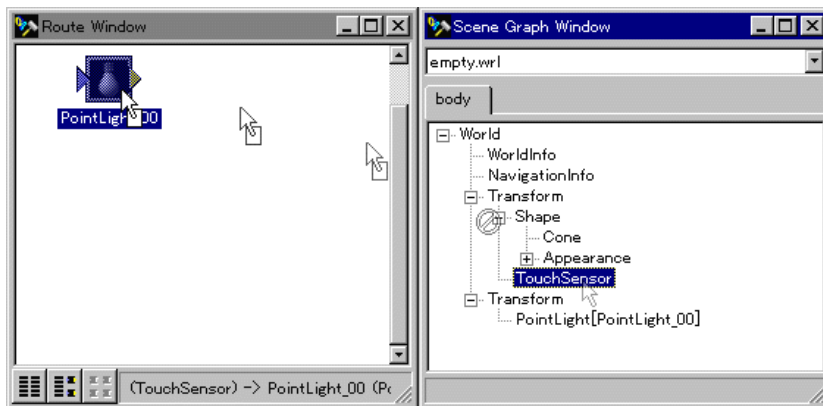


Figure 25: TouchSensor in the Scene Graph Window. It is dragged and dropped on the PointLight icon in the Route Window.

The route selection menu appears on the screen. Select "(TouchSensor).isActive TO PointLight\_00.on" out of the menu.

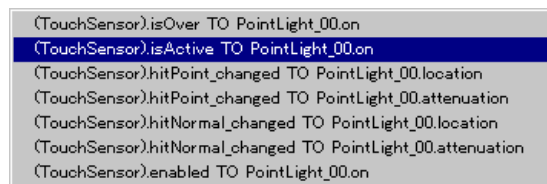


Figure 26: Route selection pop-up menu

The TouchSensor icon as well as the route between TouchSensor and PointLight appear in the Route Window.

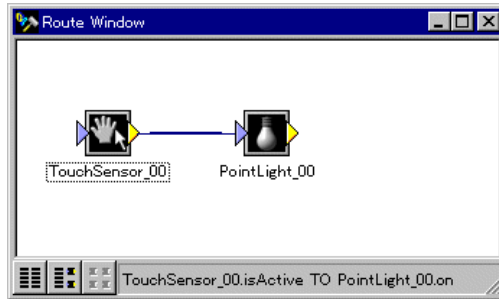


Figure 27: The route created between TouchSensor and PointLight

You have successfully created the route. You can see that the route you have created in this chapter is the same as that you did in Chapter 1.

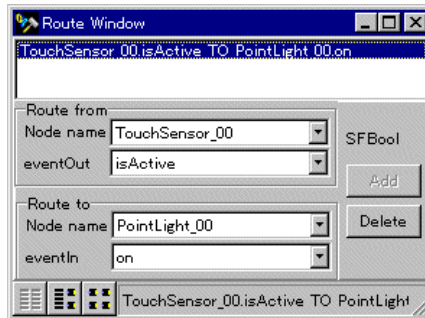


Figure 28: Displaying the route in the same display mode as Chapter 1

## 4.6 Testing the operation

Press Play and click the Cone. The Light should be turned on to illuminate the Cone.

## Keyframe Animation

You can create keyframe animation in VRML 2.0 by using a combination of nodes such as the TimeSensor node and the Interpolator node. In the next exercise, you are requested to provide keyframes used as the key points for animation. The frames between the keyframes will automatically be interpolated by VRML 2.0, which enables you to easily create natural and smooth animation.

- 5.1 Creating a Geometry node (Cone)
- 5.2 Creating and Naming a TimeSensor Node
- 5.3 Animation Edit Mode
- 5.4 Move: Creating the First Keyframe
- 5.5 Move: Creating the Second Keyframe
- 5.6 Testing the Operation
- 5.7 Changing the Speed of the Animation
- 5.8 Rotate: Creating the First Keyframe
- 5.9 Rotate: Creating the Second Keyframe
- 5.10 Testing the operation

Each step is explained in the following sections.

### 5.1 Creating a Geometry node (Cone)

First create an object. Start Conductor, open an empty world by choosing File/New, and then create a Cone.



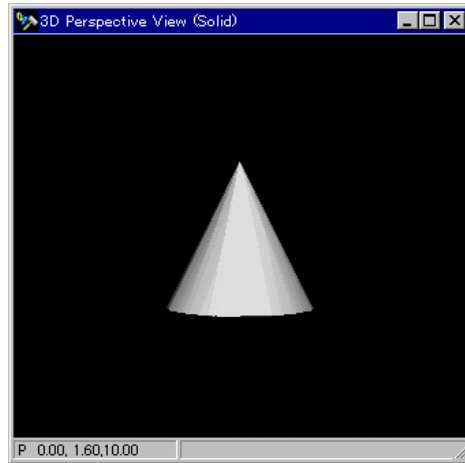


Figure 29: Creating a Cone

## 5.2 Creating and Naming a TimeSensor Node

Next, let's create a TimeSensor to be used for animation. The forms of objects in animation change as time passes by. TimeSensor is used to generate events as time passes.

First select *World* in the Scene Graph Window. Select the TimeSensor icon from the Sensor tab in the Main window, and click on the 3D View window. Then you need to name the TimeSensor. You can do this by dragging TimeSensor in the Scene Graph Window to the Route Window. It is automatically named "TimeSensor\_00" in this example.

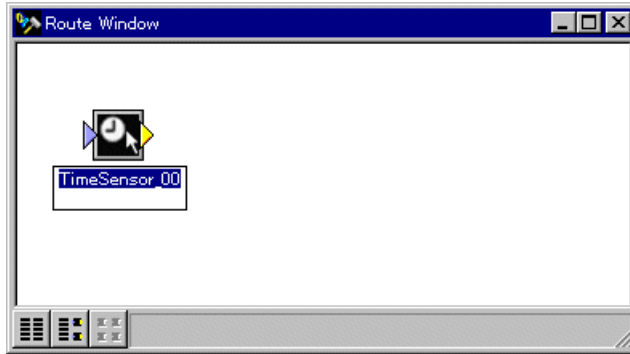



Figure 30: Naming TimeSensor by using the Route Window

### 5.3 Animation Edit Mode

Now you are ready to create keyframes. First click the animation button  in the Main window to display the Keyframe Editor window. The Keyframe Editor window enables you to create/select keyframes, and to test how animation appears on the screen.

First set necessary items in the Keyframe Editor window. Select `TimeSensor_00` in the Source field, which allows the TimeSensor created in 5.2 to be used as a timer for animation. Then, check the *Loop check box* below to loop the animation.

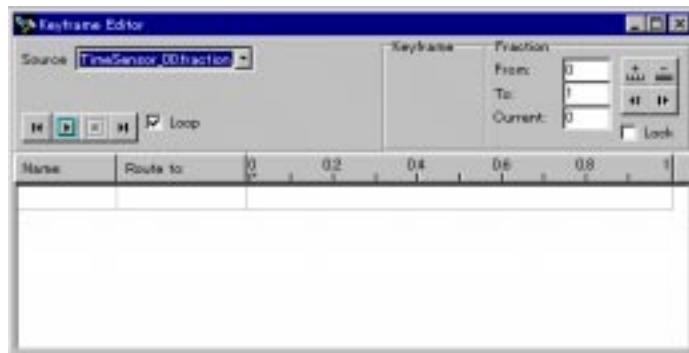


Figure 31: Set necessary items in the Keyframe Editor window

## 5.4 Move: Creating the First Keyframe

First right-click on the 3D View window to display the pop-up menu, and choose MouseMode/Move in the menu. Now you are ready to move the object with the mouse. Try moving the Cone to the upper-left of the window.

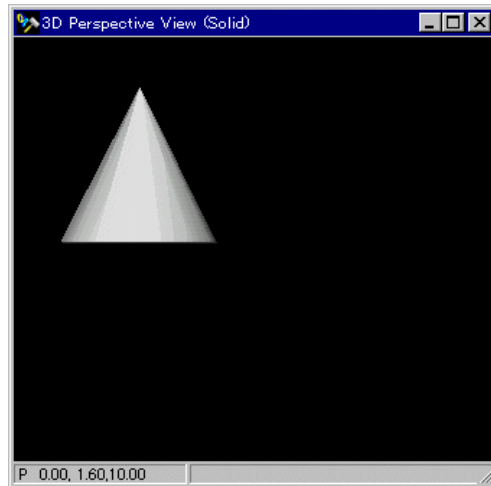


Figure 32: Moving the Cone to the upper-left of the window

The Keyframe Editor window will appear as shown on Figure 33 . The graph, located below the scale at the lower-right of the window, indicates the location where the keyframe is registered. The keyframe you have just created is registered in the scale location of 0. The locations where keyframes are registered are indicated with thick vertical lines. Keyframes should be registered within the range of 0 to 1 in general.

You can also configure the keyframes by directly setting the translation value of the Transform node in the Attribute window.

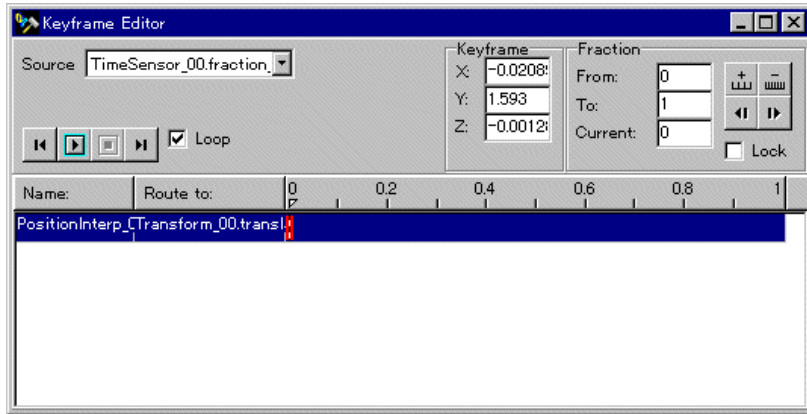


Figure 33: Registering keyframes in the Keyframe Editor window

The Route Window will appear as shown on Figure 34 . The Interpolator node (the PositionInterpolator node in this example) has automatically been created; the route from TimeSensor to PositionInterpolator to Transform has been created as well.

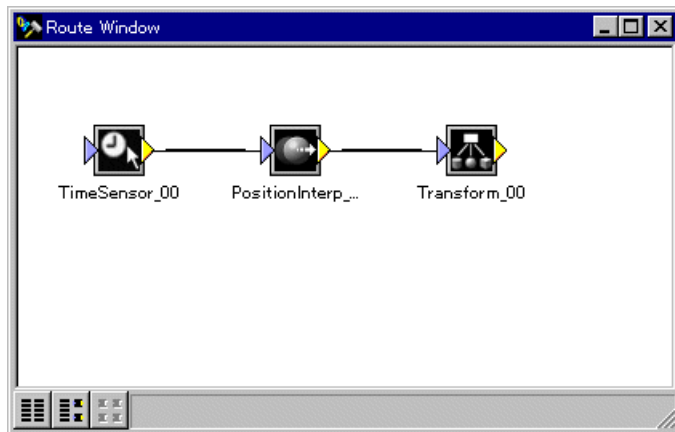


Figure 34: The Interpolator and the route newly added to the Route Window

The interpolator node creates frames out of the pre-set keyframes, through its interpolation process. It creates frames according to the time data received from the TimeSensor, and modifies the Transform node.

## 5.5 Move: Creating the Second Keyframe

Let's create the second keyframe.

Click on the 1 position of the scale in the Keyframe Editor window, to move the dotted-line locator to the position. Right-click the same position to display the pop-up menu. Choose New Keyframe on the menu to create a new frame. Check to see if a thick vertical line has newly been added between the first and second keyframes.

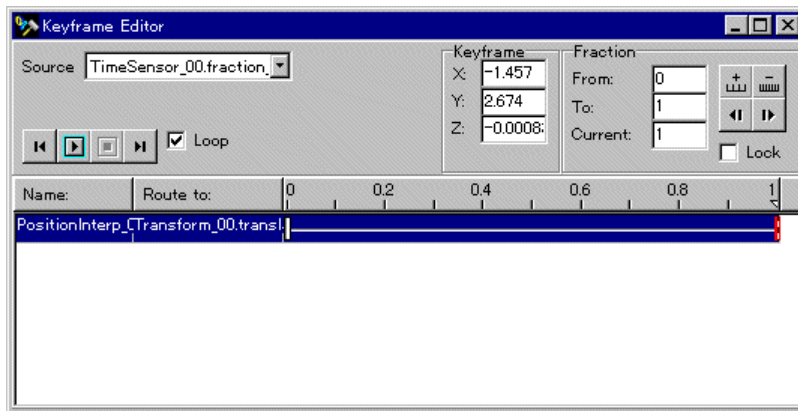


Figure 35: Creating the second keyframe in the Keyframe Editor window

Next move the Cone to the lower-right of the 3D View window. The second keyframe has been created successfully.

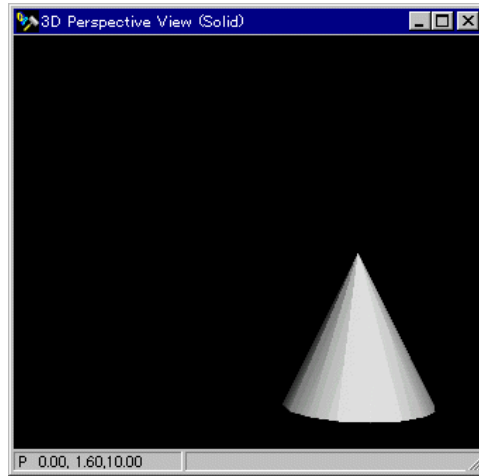


Figure 36: Moving the Cone to create the second keyframe

## 5.6 Testing the Operation

Let's play the animation by pressing the play button in the Keyframe Editor or Main window. The Cone should repeatedly move from the upper-left to the lower-right of the window. If the animation does not work properly, check in the Keyframe Editor window to see if the Loop checkbox is checked, and that each keyframe is registered correctly, and so on.

Note:1) The play button in the Main window cannot be used while Keyframe Editor window is displayed.

Note:2) To verify the movement of the overall world, end the Keyframe Editor and press the Play button on the Main window.

## 5.7 Changing the Speed of the Animation

You can change the speed of the animation by modifying the value in the `cycleInterval` field of the `TimeSensor` node. The animation presented in 5.6 seems a little too fast. Let's lower the speed here. Select `TimeSensor` in the Route Window or the Scene Graph Window. Next adjust `cycleInterval` in the Attribute Window. The total time required to play the animation should be specified in seconds. In this example, specify 4 seconds instead of 1. The larger the value, the slower the animation plays. The smaller the value, the faster it plays. The speed of animation should now be slower.

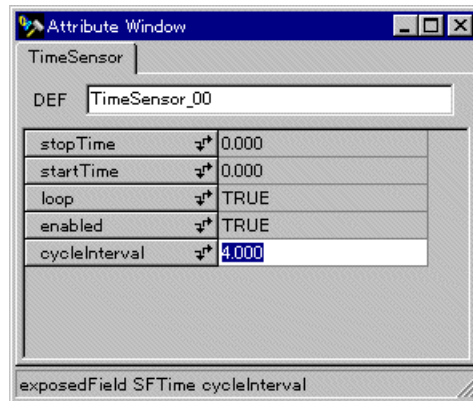


Figure 37: Changing the speed of animation with the `cycleInterval` field of `TimeSensor`

The number of frames played per second varies according to the drawing speed during run-time. Thus, smoothness of animation differs even with the same `cycleInterval` depending on its run-time environment.

## 5.8 Rotate: Creating the First Keyframe

In the previous animation exercise, the Cone only moved. Let's add rotation to it this time.

First, select the first keyframe on the graph at the left of the Keyframe Editor window. Next right-click on the 3D View window to display the pop-up menu, and choose MouseMose/Rotate on the menu. Now you are ready to rotate the Cone. Try rotating it. The job on the first keyframe is done.

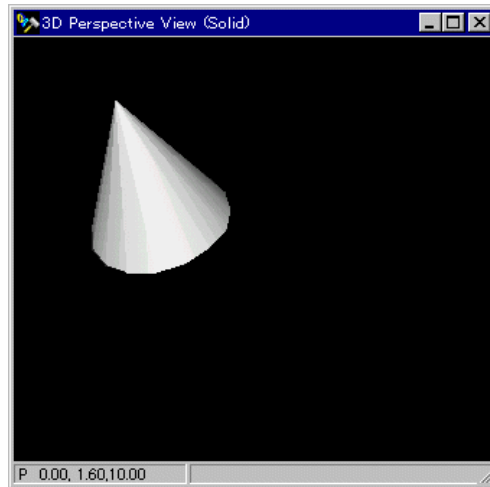


Figure 38: Rotating the Cone to create the first keyframe

After modifying the value, OrientationInterpolator is added to the Keyframe Editor window, and a new keyframe is added to the position that corresponds to the first keyframe of PositionInterpolator. OrientationInterpolator is the interpolator used to interpolate the frames for rotation. More than one Interpolator is required to attach various movements to the animation.





Figure 39: OrientationInterpolator automatically added to rotate the Cone  
OrientationInterpolator and a new route is also added to the Route Window.

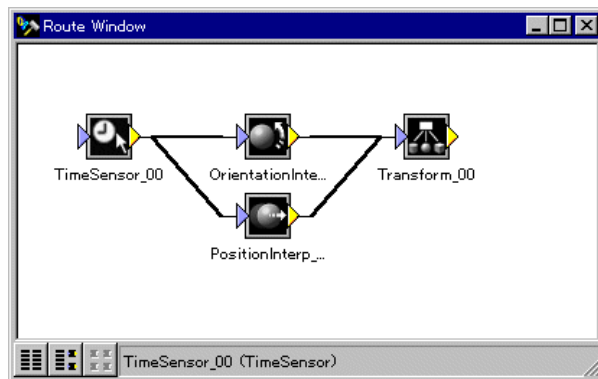


Figure 40: The icon and the new route automatically added to the Route Window

## 5.9 Rotate: Creating the Second Keyframe

Select the second keyframe by clicking the "1" position on the graph at the right of the Keyframe Editor window, in the same manner you did for the first keyframe. Then try rotating the Cone in the 3D View window.

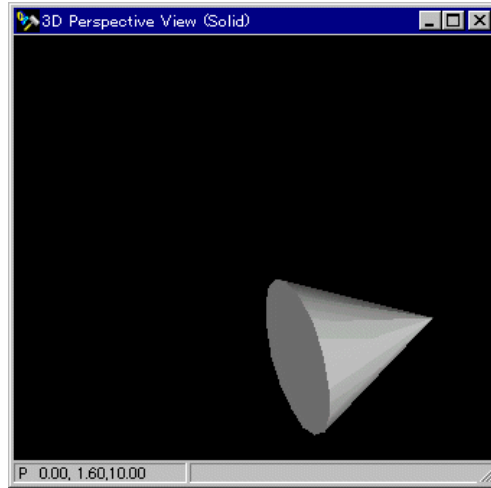


Figure 41: Rotating the Cone to create the second keyframe

Now you can see in the Keyframe Editor window that a new keyframe is added. The procedure for rotating the Cone has been completed.

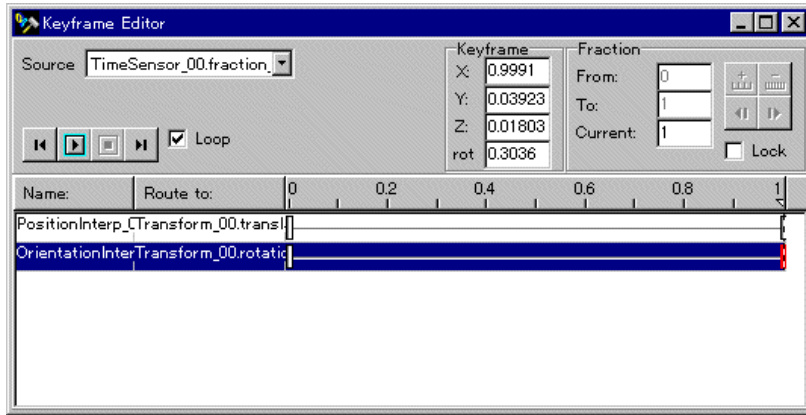


Figure 42: A new keyframe added to the Keyframe Editor window

## 5.10 Testing the Operation

Let's play the animation by pressing the play button in the Keyframe Editor or Main window as you did in 5.6. The Cone should now repeatedly rotate down from the upper-left to the lower-right corner of the window.

You can create more varied animation by incorporating changing of colors and shapes, for example. You can do this by modifying values in the `exposedField`, such as color and scale, in the Attribute Window, instead of moving or rotating the object. For details of specifying those values, see Appendix.

## PROTO Functions

Let's think about making a complicated object, such as a table, out of the number of primitive materials. If you are making more than one table, you may want to first make one table and copy it to produce more tables. What if those tables differ in color or shape in a subtle way? If the difference is small, you can register the first table as a model and reproduce other tables after the model and then add small changes to them in a simple manner using the Conductor.

The PROTO functions, newly added to VRML 2.0, enable you to specify those models. Once you create a model, you can reproduce or rearrange it as you can do with the nodes incorporated in VRML 2.0 (Cone or Cylinder, for example). You are not only allowed to group components of an object, but also to register a specified field of the component attributes as the attribute for the object newly created. Using this function, you can easily produce "Blue" or "Red" tables of the same shape, for example. You can also apply all the attributes to a new object by registering them as a single attribute. If there is more than one part used for the legs of a table, you can change the color of all the parts simply by modifying a single field. If you are familiar with object-oriented programming, you may have noticed that PROTO is similar to a Class, and reproducing an object means to produce an instance.

This chapter shows you how to use the PROTO functions through the example of creating tables. The general procedure for creating tables will be as follows:

- 6.1 Creating a table out of several primitives (Cylinder)
- 6.2 Creating PROTO
- 6.3 Reproducing a Table (Producing an Instance)
- 6.4 Showing the internal field of PROTO to the Outside (1)
- 6.5 Showing the internal field of PROTO to the Outside (2)
- 6.6 Changing the Attributes of an Instance

Each step is explained in the following sections.

## 6.1 Creating a table out of several primitives (Cylinder)

First create a table by arranging primitives.

Start the Conductor. Open a new world, and create a Cylinder. Right-click on the 3D View window to display the pop-up menu, and choose MouseMode/Move in the menu. Now you are ready to move the object with the mouse. Create and move 3 Cylinders one by one to arrange them to form a table. To change the shape of Cylinder, click Cylinder in the 3D View window and adjust its radius and height in the Attribute Window.



Figure 43: Arranging Cylinders to form a table while adjusting their shapes

Before creating Cylinders, click the parent node of Cylinder, the Transform node, in the Scene Graph Window, to let all the Cylinders become the child nodes of the same Transform node. You may also do this by cutting and pasting them with Ctrl + X or Ctrl + V keys in the Scene Graph Window after creating all of the Cylinders. If you have specified all the Cylinders as the child nodes of the same Transform node, the display should not necessarily be the same as above.

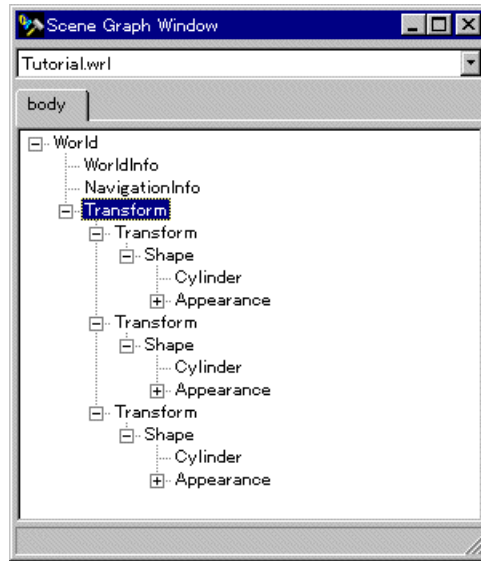


Figure 44: The Scene Graph Window after the table is produced

## 6.2 Creating PROTO

Next, let's define the combination of cylinder you have created as PROTO (a table, in this case).

Right-click the Transform node that is the parent node for all the 3 Cylinders in the Scene Graph Window to display the pop-up menu. Choose CreatePROTO on the menu, and the Input dialog appears on the screen. Type "Table" as the PROTO name.

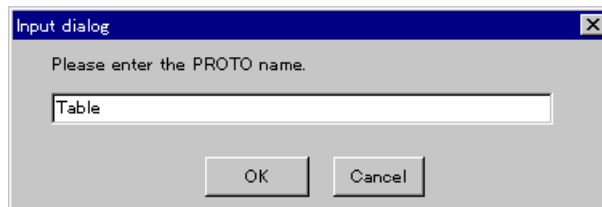


Figure 45: Typing the PROTO name in the dialog

All the attributes specified for the Transform node are combined to produce a single user-defined node "Table." The procedure for defining the PROTO has been done.

**Note:** A DEF name in capital letters will be automatically given to PROTO's when created by CreatePROTO.

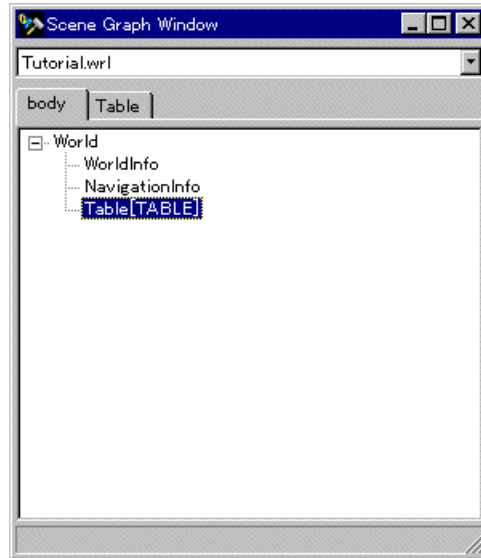


Figure 46: The Scene Graph Window after the PROTO is defined

### 6.3 Reproducing a Table (Producing an Instance)

Let's reproduce the table this time.

Right-click on the 3D View window to display the pop-up menu, and choose MouseMode/Move in the menu. Then move the table to the upper-left of the window. If necessary, move your eye position/direction further back, by pressing the down-arrow key so that the scene appears as shown on Figure 47 .



Figure 47: Moving the table

Right-click the Transform node that is the parent node for all the 3 Cylinders in the Scene Graph Window to display the pop-up menu. Choose *Add ProtoInstance/ Table* on the menu. A new table is produced on the same position as the model table. To see both tables, select the reproduced table in the Scene Graph Window and move it.

Thus, once you have defined the PROTO, you can reproduce as many tables as possible, and move or rotate them as you like.



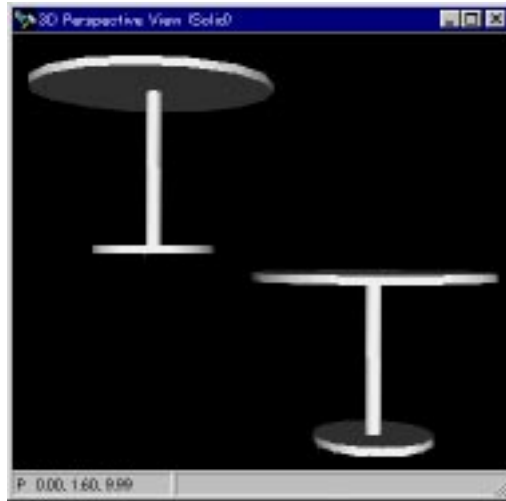


Figure 48: Creating a new table

## 6.4 Showing the internal field of PROTO to the Outside (1)

Next, let's try to change the color of each table instance. You can do this by registering the corresponding field of the internal node of Table as the field of the Table node itself, so it can be seen from the outside. Let's divide the table into several parts, such as legs, the support, and the upper board. Then register the fields used to specify the color of Cylinders as the fields for the Table node itself. Select necessary fields out of the fields of several nodes that were produced by using the PROTO function, and register them as the fields for the new node. This enables you to change the attributes of each instance, such as color or shape.

By clicking the Table tab in the Scene Graph window, the internal tree structure of Table will appear on the screen.

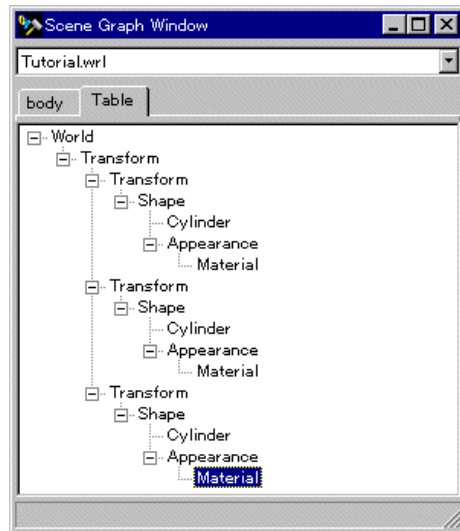


Figure 49: Displaying the internal tree structure of Table in the Scene Graph Window

Click Material of the upper board, and open the Attribute Window. If you have clicked on the Transform of the correct Cylinder node, the corresponding object should have been selected in the 3D View window.

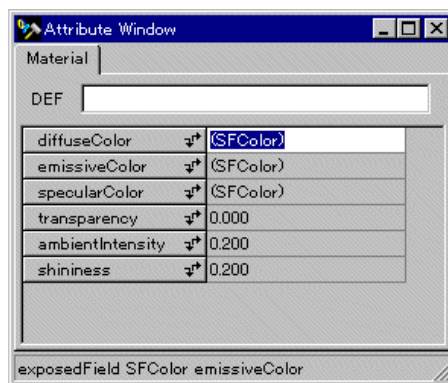


Figure 50: The attributes of the upper board of the Table

Right-click `diffuseColor` in the Attribute Window to display the pop-up menu. Choose map with IS/new on the menu, and the Add field dialog appears on the screen. Type "BoardColor" in the field name. Leave both usage and Data type in their default values: `exposedField` and `SFColor`.

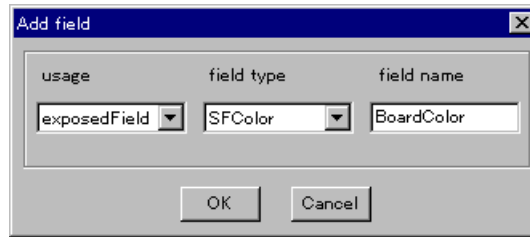


Figure 51: Typing the field name in the Add field dialog

Clicking OK in the Add field dialog, "IS BoardColor" appears in the `diffuseColor` field in the Attribute Window to indicate the registration has been completed successfully. The `diffuseColor` field of the Cylinder corresponding to the upper board of the Table can be seen as the BoardColor field of the Table newly produced, which enables you to modify it. Thus, you can change the color of the upper board of the Table by modifying the BoardColor field of the Table.

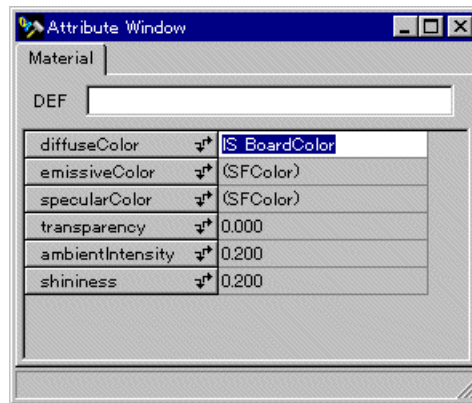


Figure 52: The Attribute Window after registering the BoardColor field

## 6.5 Showing the internal field of PROTO to the Outside (2)

Let's register the color field for the support and legs of the table as the field under for the Table node. The important point here is to map 2 different fields for the support and the legs in the same field for the Table.

Click Material of the table support Cylinder in the Scene Graph Window to open the Attribute Window. Right-click `diffuseColor` in the Attribute Window to display the pop-up menu. Choose map with IS/new on the menu, and the Add field dialog appears on the screen. Type "LegColor" in the field name. (The same procedure taken in 6.4.) The color fields for the table support and leg have been registered as the field for the Table.

## 6.6 Changing the Attributes of an Instance

Let's change the color of the table by using the field registered.

Click the body tab in the Scene Graph Window, and select the Table node (either one of them). Two fields, `BoardColor` and `LegColor`, should appear in the Attribute Window. These fields are now handled as the fields in the Table node.

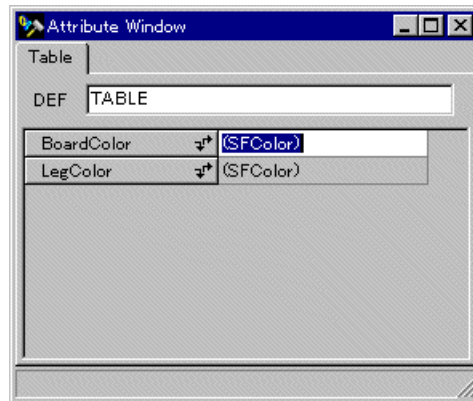


Figure 53: The registered fields displayed in the Attribute Window of the Table node

Double-click these fields to specify color in the ColorMap dialog. You can now specify the color you like to each table.

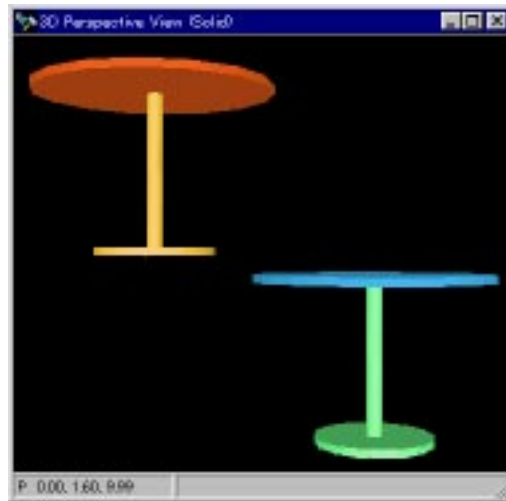


Figure 54: Specifying a color to each table

Thus, you can define the node you like by using the PROTO functions.

## **Applicable Keyframe Animation**

You have created the animation where the Cone moves while rotating in Chapter 5. This chapter shows you more advanced ways of creating animation by changing color and shape with the `exposedField` in the Attribute Window or by playing the animation you have created with `TouchSensor`. The general procedure will be as follows:

- 7.1 Creating a Geometry node (Cone)
- 7.2 Creating a TimeSensor Node
- 7.3 Animation Edit Mode
- 7.4 Varying Colors: Creating the First Keyframe
- 7.5 Varying Colors: Creating the Second Keyframe
- 7.6 Testing the Operation
- 7.7 Varying Shapes: Creating the First Keyframe
- 7.8 Varying Shapes: Creating the First Keyframe
- 7.9 Testing the Operation
- 7.10 Creating a Sensor Node (`TouchSensor`)S
- 7.11 Creating a Route
- 7.12 Testing the Operation

Each step will be explained in the following sections. Steps 7.1 through 7.3 are the same as 5.1 through 5.3 in Chapter 5, and will be skipped. Try steps 5.1 through 5.3 beforehand.

## 7.4 Varying Colors: Creating the First Keyframe

This chapter shows you the procedure for creating the animation where colors vary as time passes by. Use the Attribute Window to specify the color of the Cone. First click Material of the Cone in the Scene Graph Window, and double-click the diffuseColor field in the Attribute Window to display the ColorMap dialog.

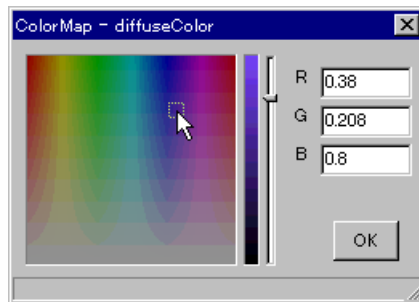
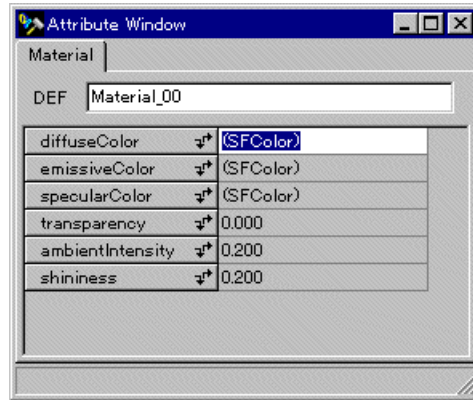


Figure 55: The diffuseColor field (Top), and the ColorMap dialog (Bottom) displayed by double-clicking diffuseColor

A bluish color is specified here.

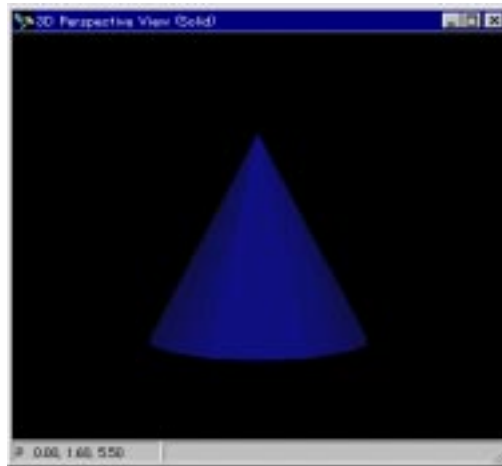


Figure 56: The bluish color specified for the first keyframe

A new ColorInterpolator has been added to the Keyframe Editor window, and the selected color appears in the Keyframe field at the upper-center of the Keyframe Editor window. The Keyframe field displays various information according to the type of Interpolator used. You may find this information very helpful when editing keyframes.

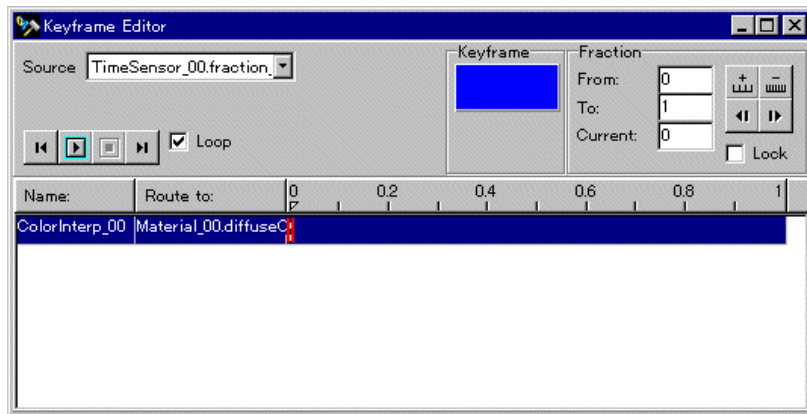


Figure 57: The Keyframe Editor window after the keyframe has been registered



In the Route Window, the ColorInterpolator icon, the Material icon, and a new route is added. The ColorInterpolator received time information from TimeSensor, generates the color corresponding to the data, then passes it to Material. There are various types of Interpolator nodes, such as Position, Orientation, and Color, to meet the interpolation needs for each animation.

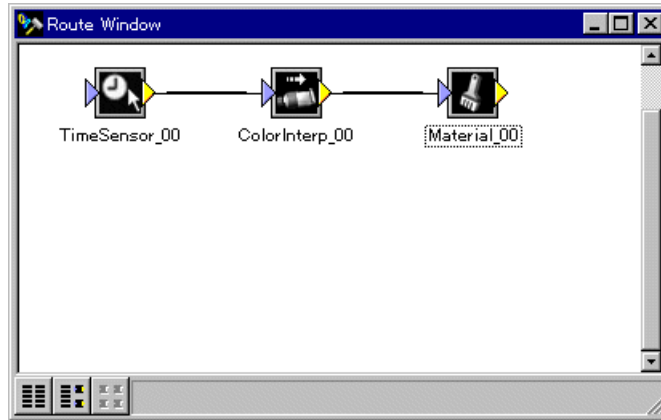



Figure 58: Icons and a route automatically added to the Route Window

Interpolator node and the route are automatically created to allow various types of animation. This time, you have used the `diffuseColor` field, but you can use other fields as well. You can use, however, only the `exposedFields` which only allow you to edit items from the Interpolator node. If the field is not an `exposedField`, it cannot be manipulated from the Interpolator node, and thus cannot be used for animation. Fields that have  after their field names in the Attribute window are the `exposedFields`.

## 7.5 Varying Colors: Creating the Second Keyframe

Let's create the second keyframe.

Click on the 1 position of the scale in the Keyframe Editor window, to move the dotted-line locator to the position. Right-click the same position to display the pop-up menu, and choose New Keyframe on the menu to create a new frame. Double-click the `diffuseColor` field in the Attribute Window as you have done in 7.4. Then specify a reddish color in the ColorMap dialog.

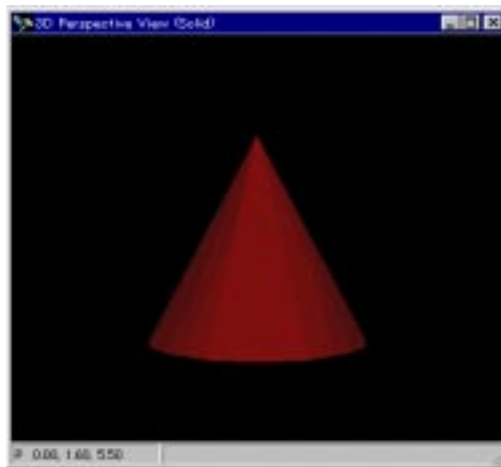


Figure 59: The reddish color specified for the second keyframe

The Keyframe Editor window appears as follows:

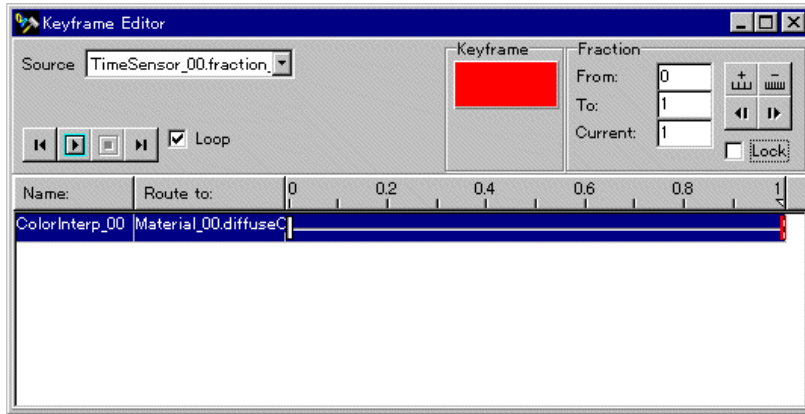


Figure 60: The Keyframe Editor window after the keyframe has been registered

## 7.6 Testing the Operation

Let's play the animation you have created. Click the play button in the Keyframe Editor window or the Main window. The color of the Cone repeatedly varies from blue to red. If the animation is too fast, adjust its speed by using the `cycleInterval` field of `TimeSensor` (see 5.7).

## 7.7 Varying Shapes: Creating the First Keyframe

You can create various types of animation in the animation edit mode just by specifying items in the Attribute Window. Next, let's create an animation where the shape of an object varies as its color varies.

Thus, editing items in the Attribute Window in animation edit mode, select the first keyframe on the graph at the right of the Keyframe Editor window. Then click the parent node of the Cone, the Transform node, in the Scene Graph Window to display the Attribute Window. Click the scale field (after making sure it is an `exposedField`) in the Attribute Window. The scale field is used to specify the enlarge/reduce ratio of an object in the directions of x, y, and z. Clicking the field, the values currently set to x, y, and z will displayed. You can edit these values as you like.

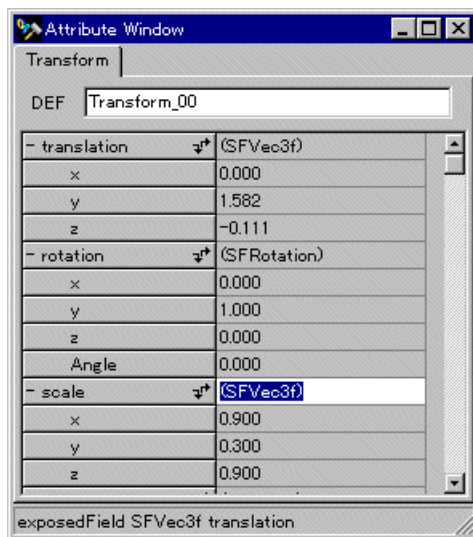


Figure 61: The scale field to be modified

The object has been a little flattened here.



Figure 62: The Cone, a little flattened by adjusting the scale field

You can see that PositionInterpolator has been added in the Keyframe Editor window.

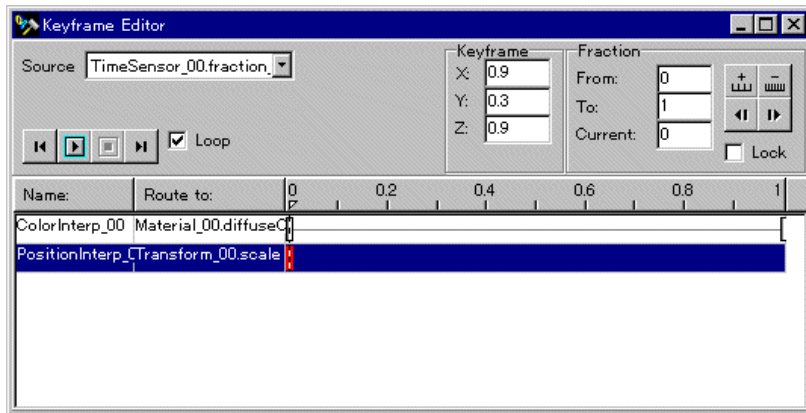


Figure 63: The Keyframe Editor window after the keyframe has been registered

You can also see in the Route Window that the PositionInterpolator node, the Transform node icon, and a route is created.

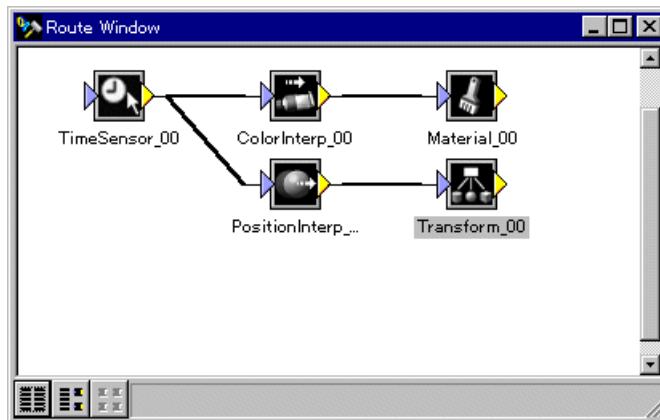


Figure 64: Icons and a route automatically added to the Route Window

## 7.8 Varying Shapes: Creating the First Keyframe

Click on the 1 position of the scale in the Keyframe Editor window, to move the dotted-line locator to the position. Right-click the same position to display the pop-up menu, and choose New Keyframe on the menu to create a new frame. Then edit the scale field at the upper-center of the Attribute Window or the Keyframe Editor window. The object has been made a little narrower.



Figure 65: The Cone a little narrowed by adjusting the scale field

The Keyframe Editor window appears as follows:

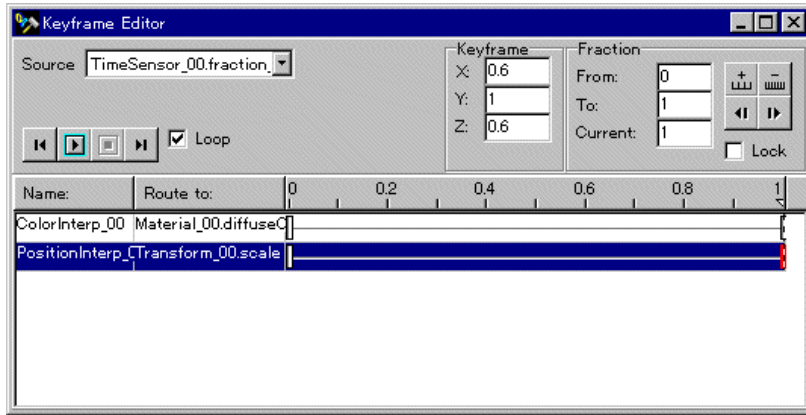


Figure 66: The Keyframe Editor window after the keyframe has been registered

## 7.9 Testing the Operation

Let's play the animation you have created. Click the play button in the Keyframe Editor window or the Main window. The Cone repeatedly varies its shape from flat to the narrow.

## 7.10 Creating a Sensor node (TouchSensor)

Finally let's get the animation to be played responding to the switch operation. The animation you have created so far was simple, but you can create more advanced animation by using a switch where an automatic door opens responding to the switch operation, for example.

This time, let's play the animation when the Cone is clicked. First, set the Loop checkbox to off, then create a TouchSensor. Select the parent node of the Cone, the Transform node, and add a TouchSensor.

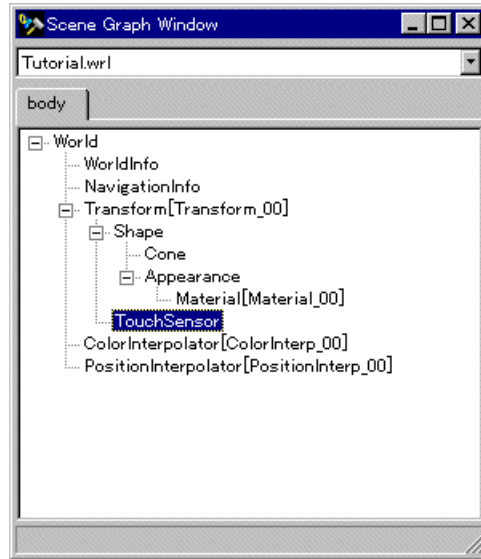


Figure 67: Registering TouchSensor as the child node of the Transform node

## 7.11 Creating a Route

Drag and drop TouchSensor in the Scene Graph Window onto the TimeSensor icon in the Route Window. Then choose "TouchSensor\_00.touchTime TO TimeSensor\_00.startTime" on the pop-up menu. This means that you specify the time the TouchSensor (TouchSensor\_00) was pressed (touchTime) as the start time (startTime) for the TimeSensor (Timsensor\_00). This enables the TimeSensor to activate by clicking TouchSensor to play the animation. You can use the same method with other Sensor nodes.

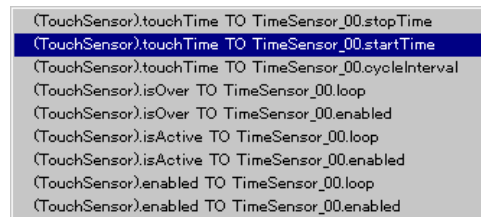


Figure 68: The pop-up menu that appears after TouchSensor dragged



New icons and routes are added to the Route Window as shown:

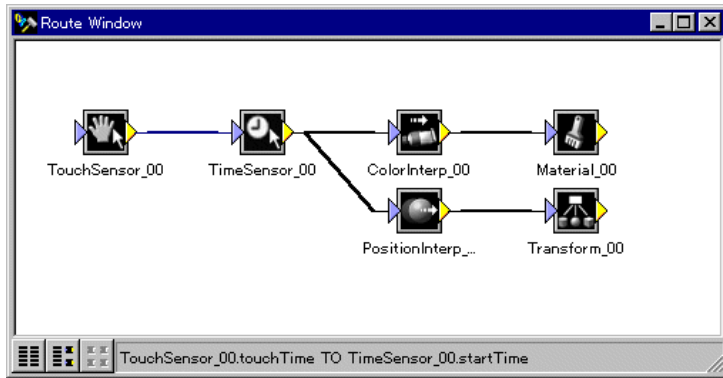


Figure 69: The Route Window after the routed added

## 7.12 Testing the Operation

Let's play back the animation. This time, press the play buttons on the Main window as the animation also includes TouchSensor. Every time you click the Cone in the 3D View window, the Cone changes its form from flat to narrow while changing its colors as well.

---

## References

---

Rodger Lea, Kouichi Matsuda, Ken Miyashita: "Java for 3D and VRML Worlds", New Riders.