

# CyberToolbox

Release 2.0

Java

## User's Guide

---

<b>What is CyberToolbox.....</b>	<b>3</b>
<b>Installation.....</b>	<b>4</b>
<b>Tutorial.....</b>	<b>5</b>
<i>Rotating ball.....</i>	<i>5</i>
1. Creating a ball object.....	5
2. Creating a diagram to rotate the object.....	6
3. Starting the simulation.....	8
<i>Flushing ball.....</i>	<i>9</i>
1. Creating a ball object with a material.....	9
2. Creating a picking event.....	10
3. Creating a diagram to initialize the object color.....	11
4. Creating a diagram to change the object color by clicking....	12
5. Starting the simulation.....	13
<b>Operation Overview.....</b>	<b>14</b>
<i>World Window.....</i>	<i>15</i>
SceneGraph Tree.....	16
Route Tree.....	17
Event Tree.....	18
ToolBar.....	20
<i>Perspective Window.....</i>	<i>22</i>
Toolbar.....	22
<i>Diagram / Module Window.....</i>	<i>26</i>
Toolbar.....	28
<b>Behavior Overview.....</b>	<b>29</b>
<i>Event.....</i>	<i>30</i>
System Event.....	30
User Event.....	31
<i>Diagram.....</i>	<i>34</i>
<i>Module.....</i>	<i>34</i>

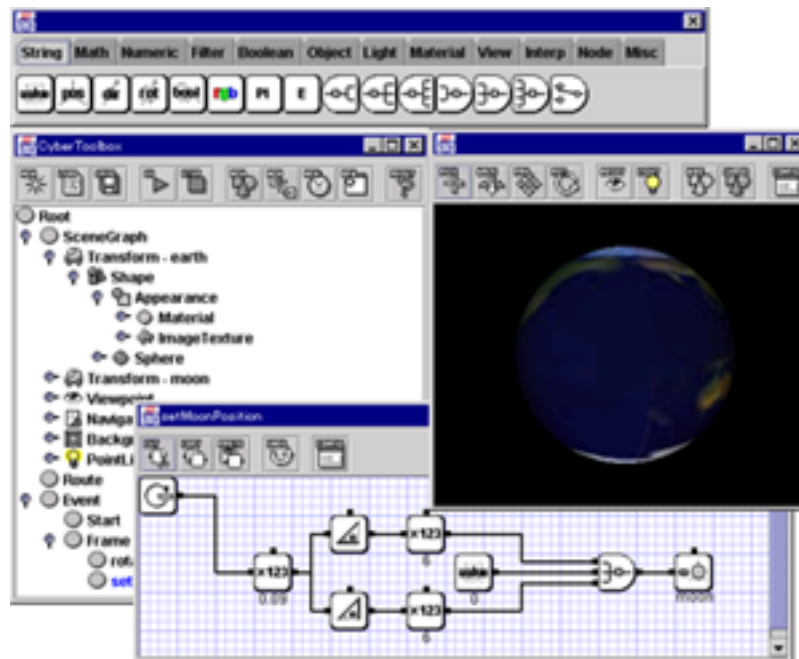
<b>Module Behavior Overview .....</b>	<b>36</b>
<i>String</i> .....	36
<i>Numeric</i> .....	43
<i>Math</i> .....	50
<i>Filter</i> .....	61
<i>Boolean</i> .....	67
<i>Geom (Geometry)</i> .....	72
<i>Object</i> .....	75
<i>Material</i> .....	78
<i>Light</i> .....	82
<i>Viewpoint</i> .....	86
<i>Interp (Interpolator)</i> .....	88
<i>Misc</i> .....	90

## ***What is CyberToolbox ?***

---

CyberToolbox is a VRML2.0/97 authoring tool for WIN32 and Java platforms. VRML is the most standard 3D file format on the World Wide Web now. However, It is difficult to create good interactive behaviors for beginner creators who have not been studied computer programming languages, because the creators have to use the languages, Java or Java Script, to create the behaviors.

CyberToolbox has a visual programming language so that the creators can create the interactive behaviors easily. Using CyberToolbox, the creators can create the behaviors only to connect between icon modules by data-flow lines.



Because a content which is created using CyberToolbox is saved into a standard VRML97 file with some class files which are used to run the behaviors, everyone can see the content using Microsoft Internet Explore or Netscape Communicator with VRML plug-ins.

I have developed CyberToolbox with CyberVRML97 that is development libraries of C++ and Java. If you have any interest in VRML application developments, you can get the information in more detail from my web, <http://www.cyber.koganei.tokyo.jp>.

## ***Installation***

---

To use CyberToolbox for Java, you have to install latest Java2 (JDK1.2) and Java3D packages. If you don't install the packages yet, get the packages from Sun's Java site (<http://java.sun.com>),...

If you have installed a VRML-Java3D package of the Java3D and VRML Working group (<http://www.vrml.org/WorkingGroups/vrml-java3d/>), you should remove the package to install CyberToolbox easily because the following class files in the VRML-Java3D package conflicts with the VRML-CyberToolbox package.

<http://www.vrml.org/Specifications/VRML97/part1/java.html#B.9>

The CyberToolbox is distributed as a jar file. To extract the package, use a jar tool that is included with JDK utility or WinZip utility. For example, to extract the package using the jar tool,

```
jar xvf ctb200.zip
```

To run the CyberToolbox, use following commands.

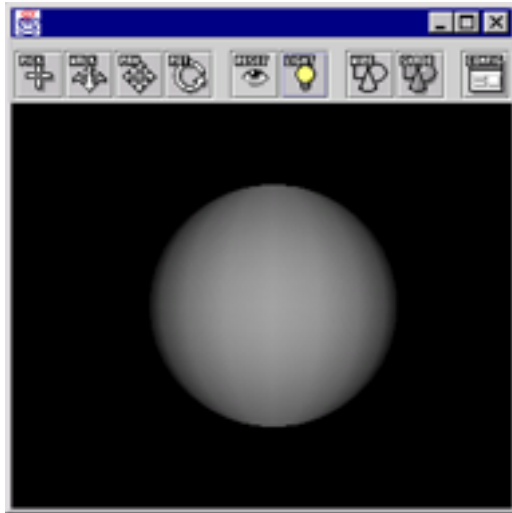
```
cd cybertool box  
run (or run.bat)
```

# ***Tutorial***

---

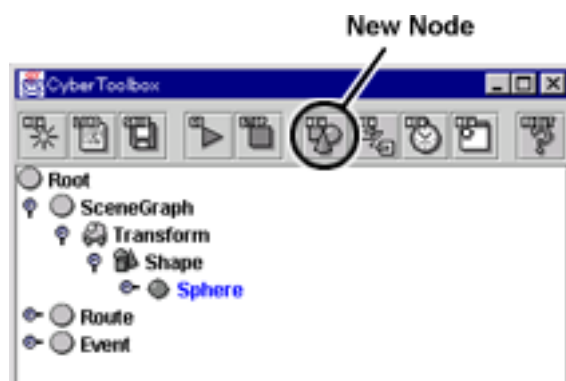
## **Rotating ball**

This tutorial shows you how to rotate a ball object using CyberToolbox. To create the content, first add the ball, next create a diagram to rotate the ball.



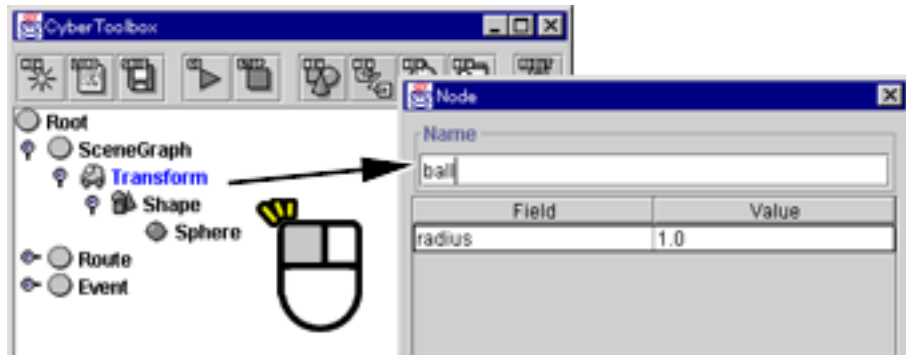
### ***1. Creating a ball object.***

To add a ball object, you have to add three nodes into a current scene-graph using a “New Node” button in World window tool bar. To create the object, add a Transform node into the scene-graph as a top node, add a Shape node into the Transform node, and add a Sphere node into the Shape node.

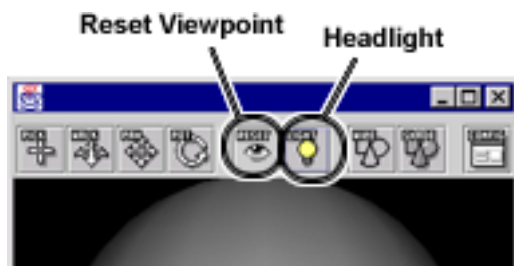


See “Operation Overview” about creating a new node in more detail.

To rotate the ball object, you have to set a name into the added Transform node. Click the node, and set the name as “ball”.

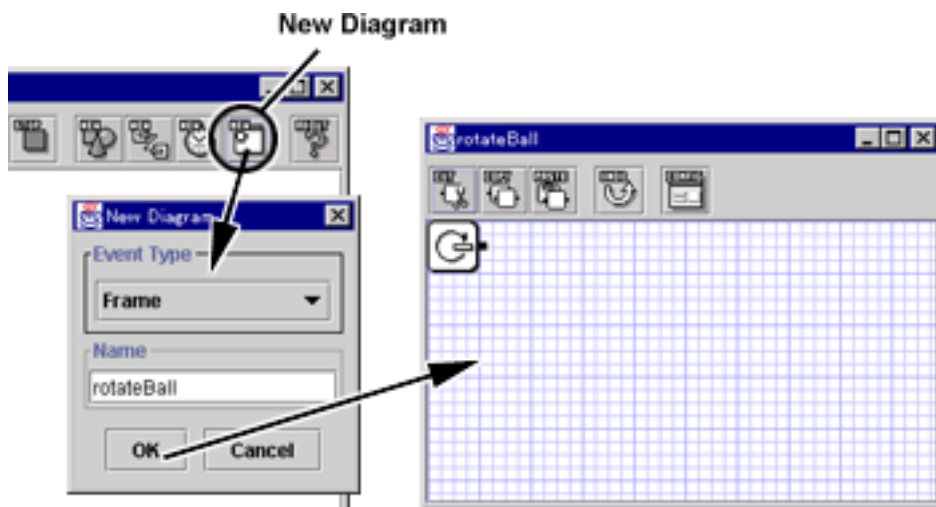


To see the added object in Perspective window, push “Reset Viewpoint ” button and select “XY Plane View”, then push “Headlight” button to turn on a headlight if the headlight is off.







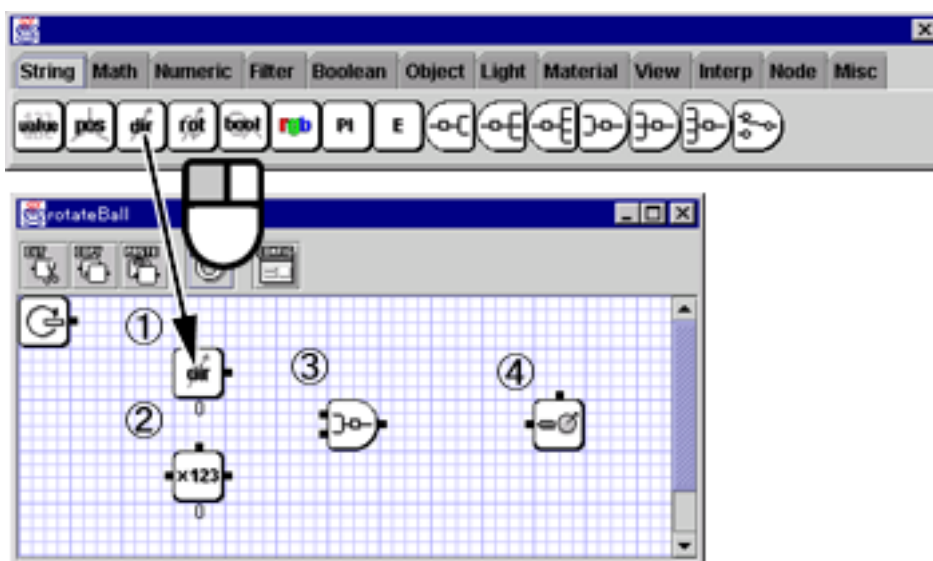
## 2.Creating a diagram to rotate the object

To rotate the added object, you have to add a diagram, and create the behavior using four modules. First create a diagram that is executed by Frame event. To add the diagram, push “New Diagram” button in World window, select “Frame” as the event type, and set “rotateBall ” as the name.

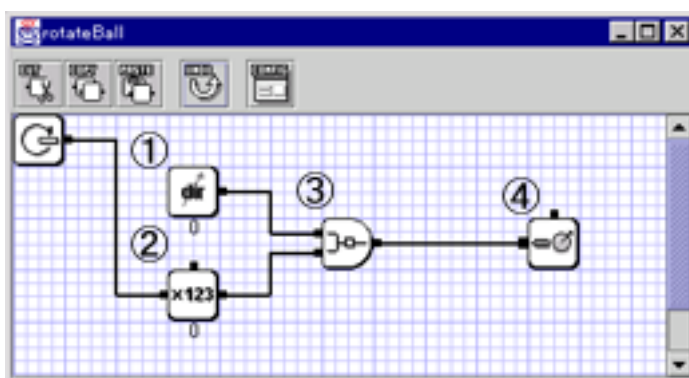


Next, drop the following four modules into the diagram from Module window,

	Icon	Class	Name
1		String	Direction
2		Filter	Scale
3		String	Mearge2Values
4		Object	SetRotation






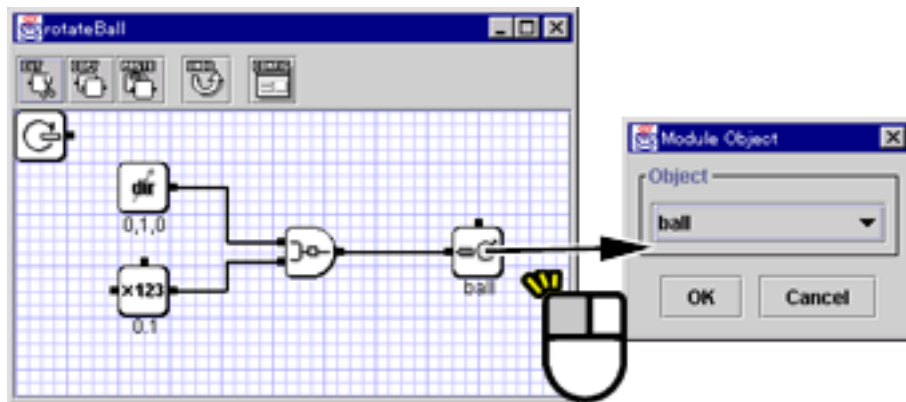
Then connect data-flow lines between the modules as following.





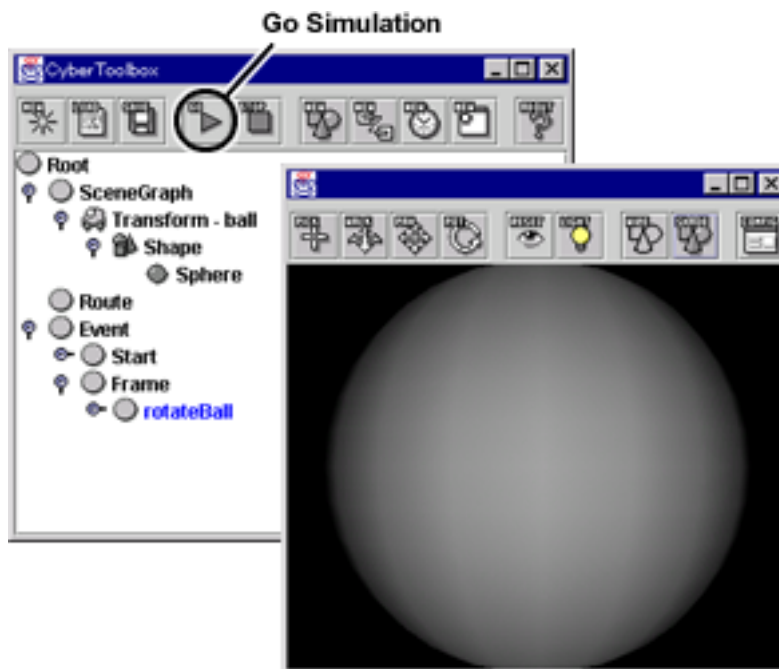
Finally, set internal values of the following module to double-click.

Icon	Setting value
	X=0, Y=1, Z=0 (0,1,0)
	0.1
	ball



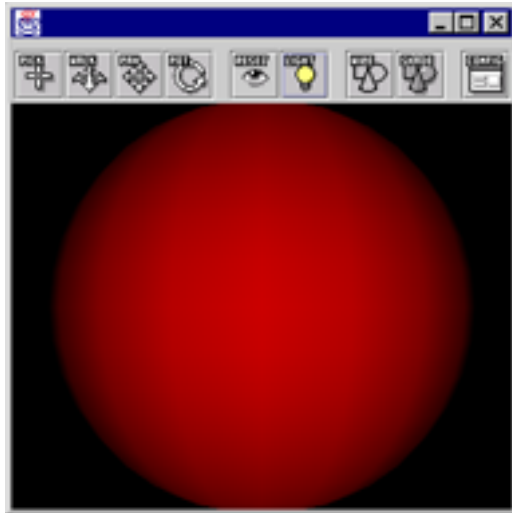
### 3. Starting the simulation

To check the behavior, click “Go Simulation” button in World window tool bar. When the simulation is active, the ball is rotated.



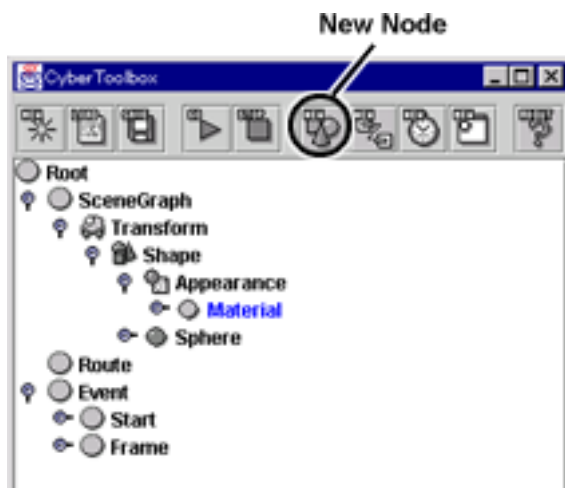
## **Flushing ball**

This tutorial shows you how to flush a ball object when the object is clicked using CyberToolbox. To create the content, first add the ball, then add the picking event, finally create diagrams to flush the ball.

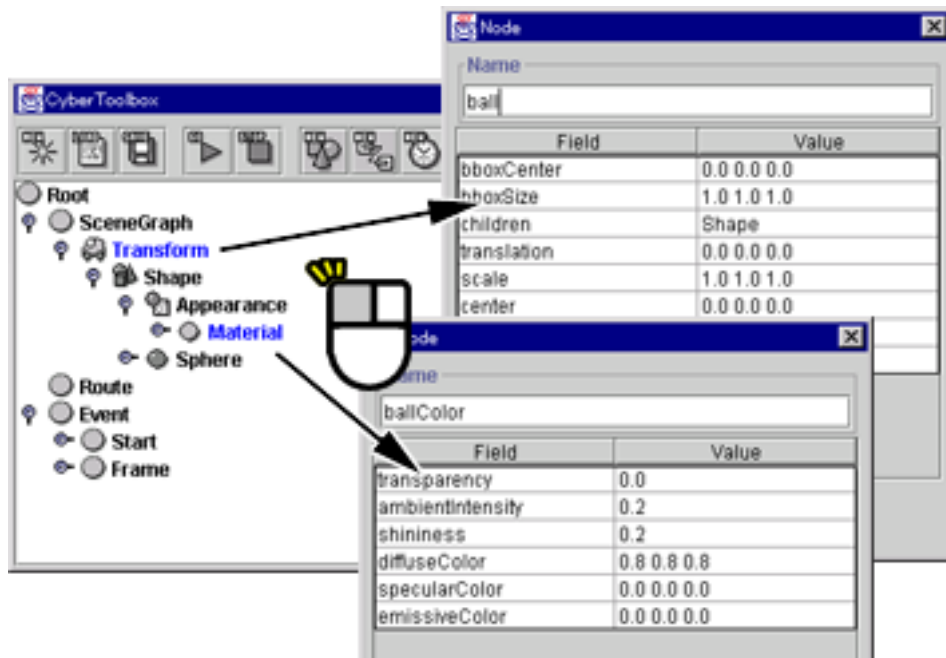


### **1. Creating a ball object with a material.**

To add a ball object, you have to add five nodes into a current scene-graph using a “New Node” button in World window tool bar. To create the object, add a Transform node into the scene-graph as a top node, add a Shape node into the Transform node, add an Appearance node and a Sphere node into the Shape node, and add a Material node into the Appearance node. .

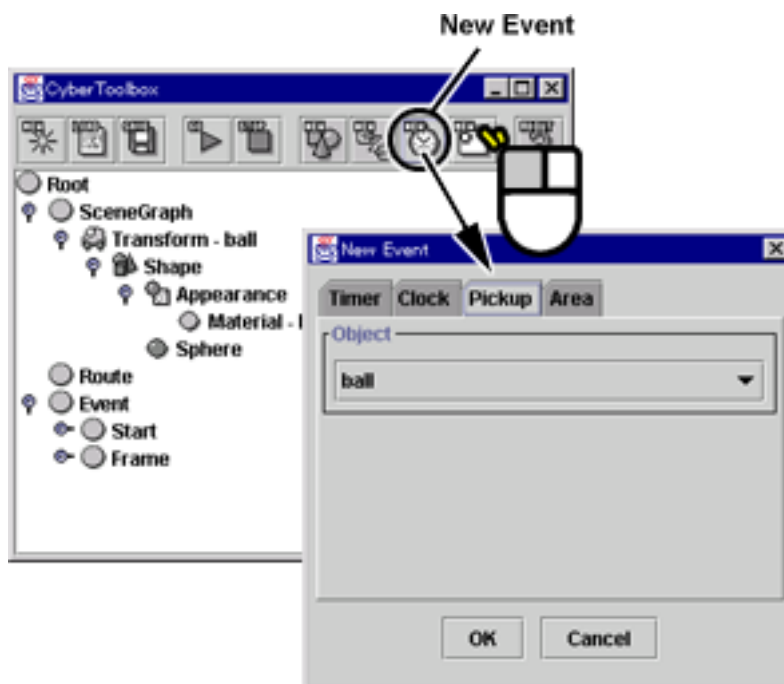


To click the ball object, you have to set a name into the added Transform node. Click the node, and set the name as “ball”. Similarly, to change the object color, you have to set a name into the added Material node. Click the node, and set the name as “ballColor”



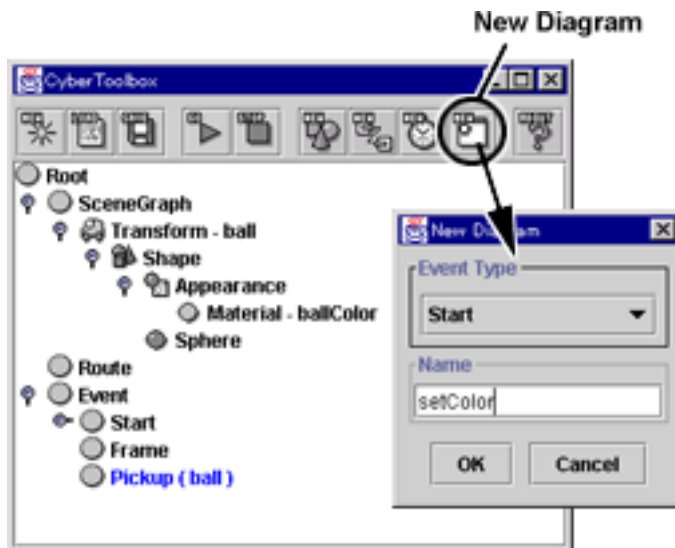
## 2. Creating a picking event.

To create this behavior, you have to add a picking event at first. To add the event, push “New Event” button in World window, select “Pickup” tab, select “ball” item in the list box, and click OK.





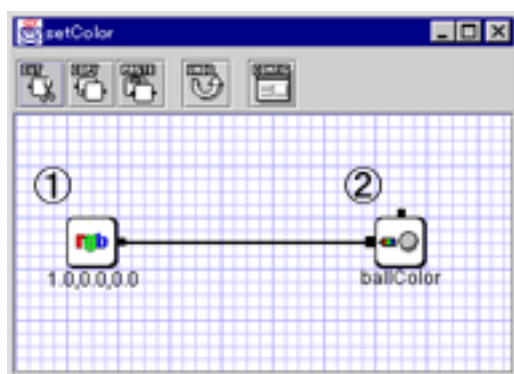
### 3. Creating a diagram to initialize the object color

To set the object color into red when simulation is started, add a diagram that is executed by Start event. To add the diagram, push “New Diagram” button in World window, select “Start” as the event type, and set “setColor” as the name.



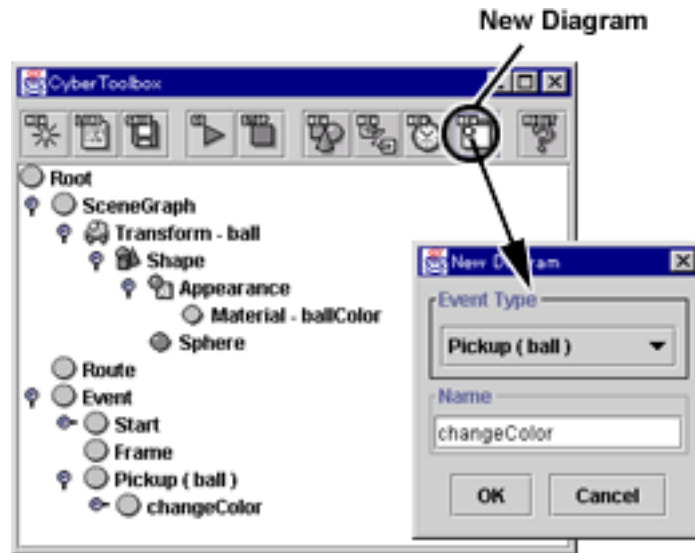
Then, add the following two module into the created diagram, set module values, connect a data-flow line.

	Icon	Class	Name	Value
1		String	Color	Red (255, 0, 0) = (1.0, 0.0, 0.0)
2		Material	SetDiffuseColor	ballColor



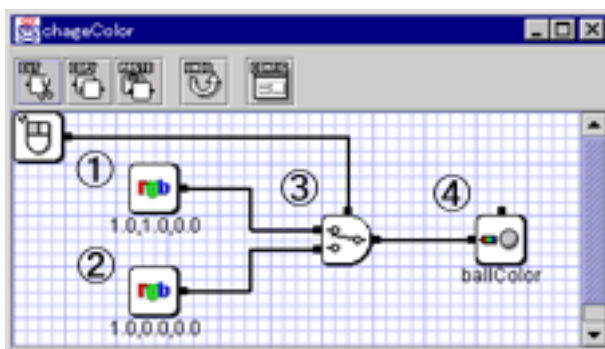
#### 4. Creating a diagram to change the object color by clicking

To set the object color into yellow when the object is clicked and set into red when the object is released, add a diagram that is executed by the picking event. To add the diagram, push “New Diagram” button in World window, select “Pickup (ball)” as the event type, and set “changeColor” as the name.



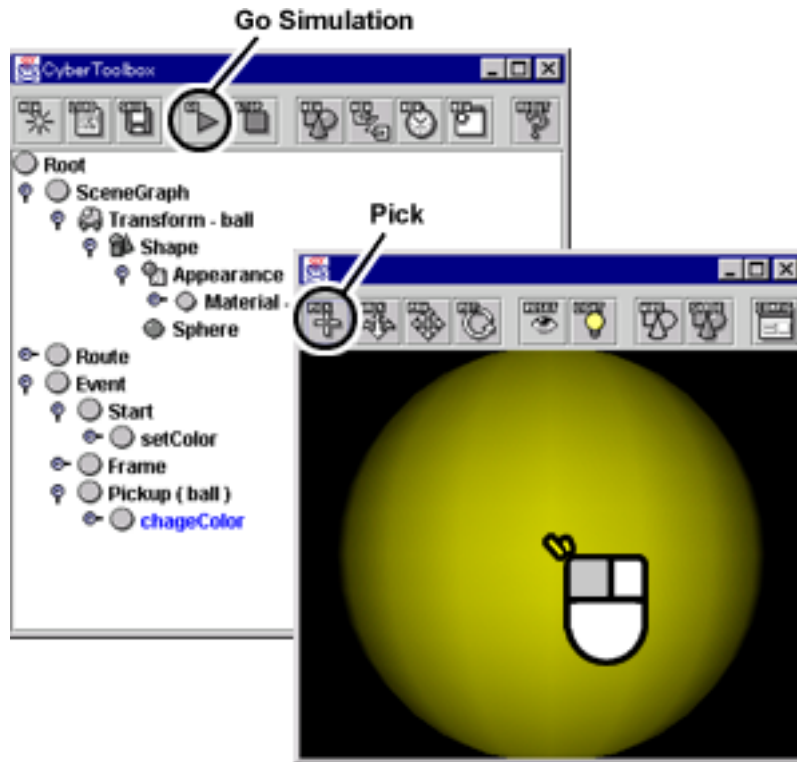
Then, add the following two module into the created diagram, set module values, connect a data-flow line.

	Icon	Class	Name	Value
1		String	Color	Yellow (255, 255, 0) = (1.0, 1.0, 0.0)
2		String	Color	Red (255, 0, 0) = (1.0, 0.0, 0.0)
3		String	Selector	
4		Material	SetDiffuseColor	ball



## 5. Starting the simulation

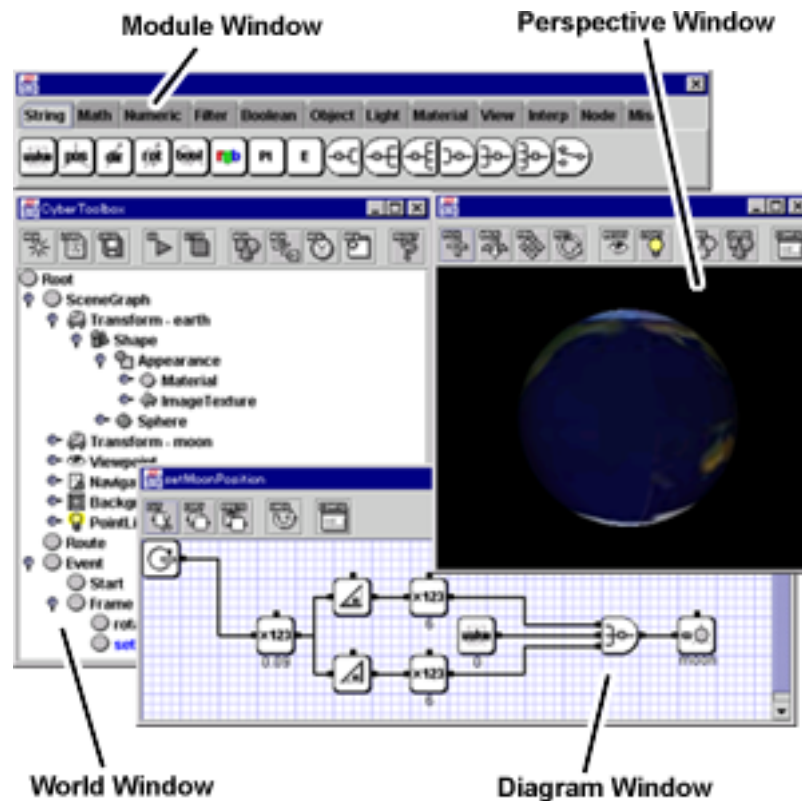
To check the behavior, click “Go Simulation” button in the World window tool bar, and click “Pick” button in the Perspective window. When you click the ball in the Perspective window, the color is changed into yellow.



## Operation Overview

---

CyberToolbox for Java has four main windows, World window, Perspective window, Diagram window and Module window.



World window shows all VRML node and route information, events and diagrams. Using the window, you can edit all VRML information visually, and add new events and diagrams to create behaviors.

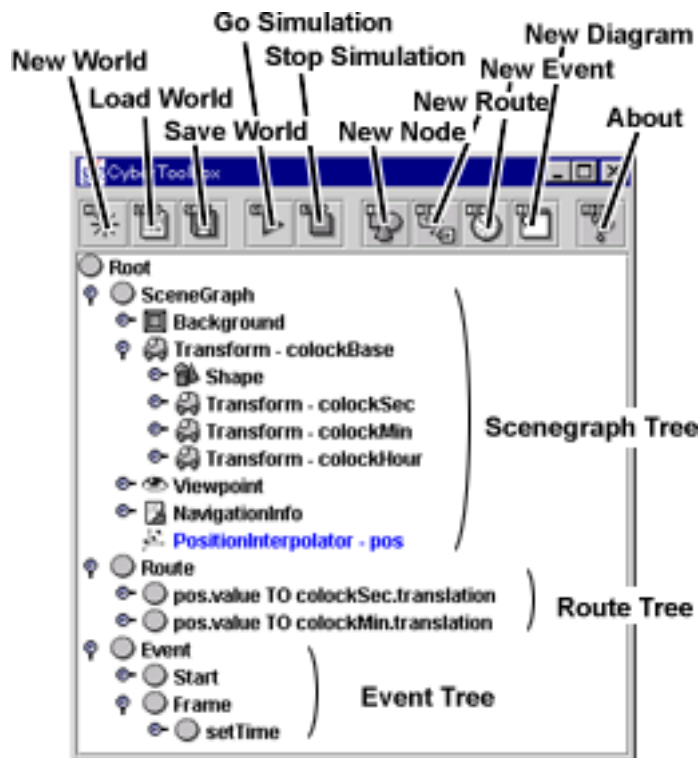
Perspective window shows graphical objects in the current world. You can check the world with the behaviors when the simulation is active, and walk in the world, and click the objects using a mouse.

Diagram window is a workspace of creating behaviors using modules in Module window, you can create the behaviors by only mouse operations.

## **World Window**

This window shows current scene-graph, event and diagram information, allows you to read a VRML file to add the scene-graph information into a current world, save the current world information into a VRML file, create new nodes, events, diagrams which are workspaces to create behaviors, start and stop the simulation.

This window has three main trees, SceneGraph , Route , and Event tree.

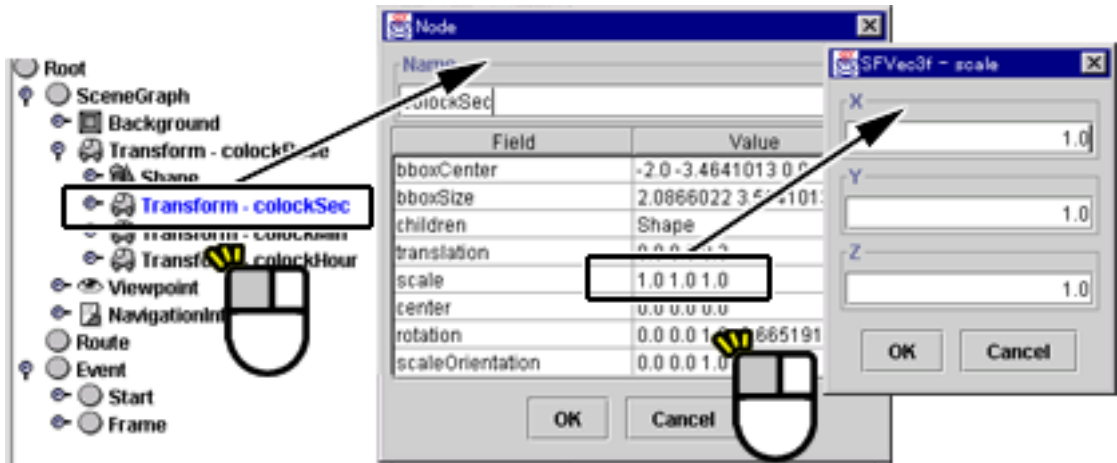


The SceneGraph tree shows all VRML node information, and the Route tree shows all VRML route information, and the Event tree shows all events and diagrams in the current world.



## SceneGraph Tree

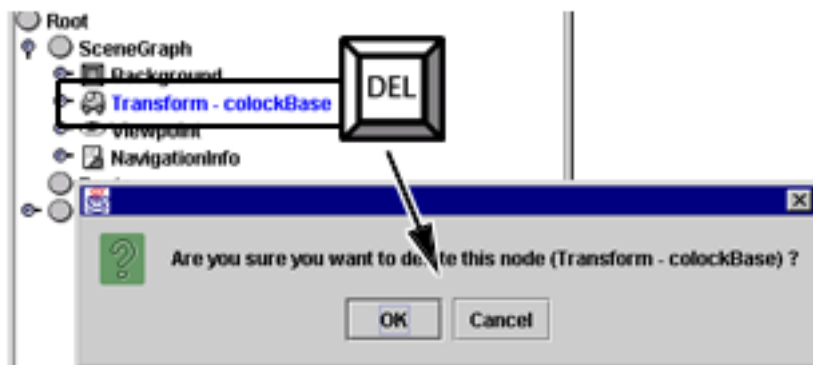
This tree shows all VRML node information in the current world. To confirm the node information in the tree, double-click the node item to open the setting dialog. To edit the field value, double-click the field in the dialog to open the dialog.



To move a node into under other parent node, drag the node item and drop on the new parent node item. If you want to move into the top of the scene-graph, drop the node on “SceneGraph” tree item.



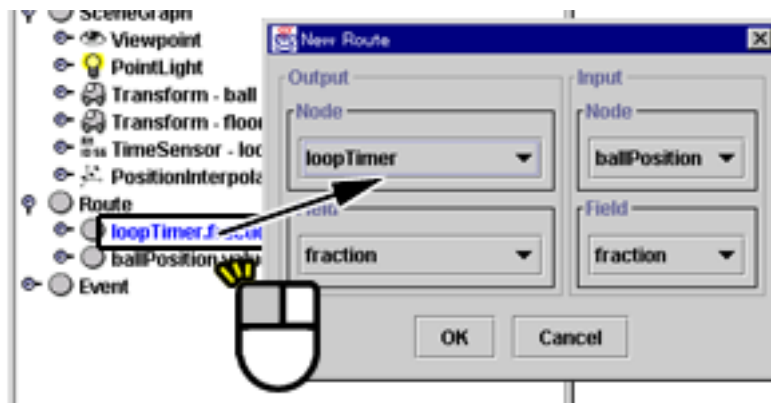
To delete a node, click the node item, then press the delete key. Click “Ok” button on the confirmation dialog if you sure want to delete it.



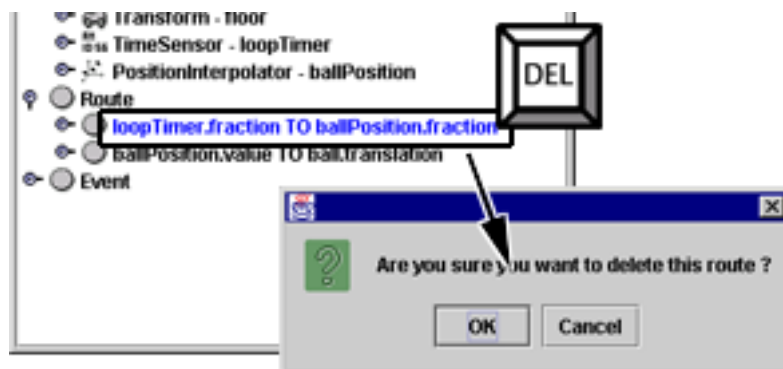
Use “New World” button on the toolbar to add a new node into the current world. See following “Toolbar” section about the button in more detail.

## Route Tree

This tree shows all VRML route information in the current world. To confirm a route information in the tree, double-click the route item to open the setting dialog.



To delete a node, click the node item, then press the delete key. Click “Ok” button on the confirmation dialog if you sure want to delete it.

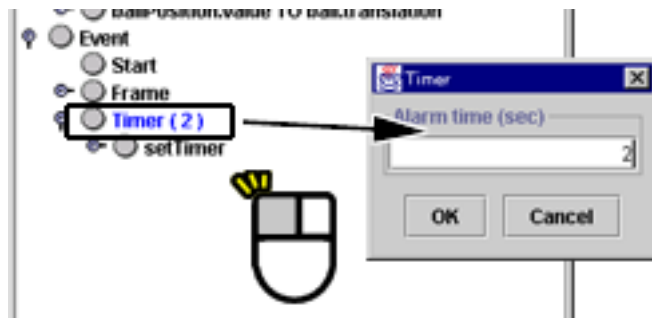


Use “New Route” button on toolbar to add a new route into the current world. See following “Toolbar” section about the button in more detail.

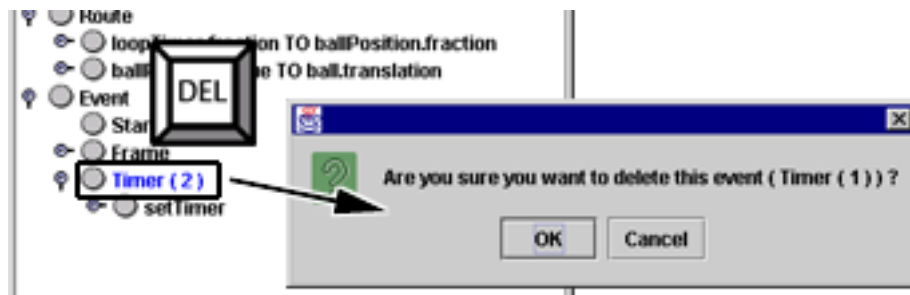
## Event Tree

This tree shows all events and diagrams in the current world. The event is first trigger to run behaviors, and the event run the related diagrams. See “Behavior Overview” section about the event and diagram in more detail.

To edit options of a user event that is added by a user operation, double-click the event item to open the setting dialog. Note that you can’t edit system events that are added by CyberToolbox at first. Use “New Event” button on the toolbar to add a new user event into the current world. See following “Toolbar” section about the button in more detail.

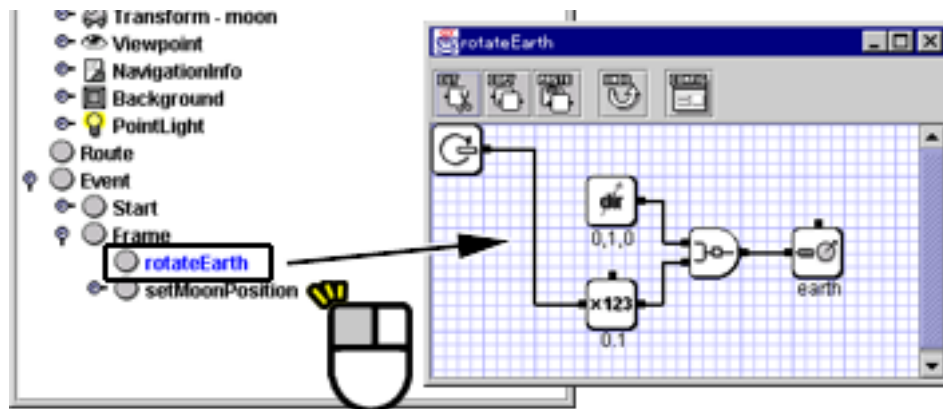


To delete an event, click the event item, then press the delete key. Click “Ok” button on the confirmation dialog if you sure want to delete it. When a event is deleted, the related diagrams are deleted too.



Use “New Event” button on the toolbar to add a new event into the current world. See following “Toolbar” section about the button in more detail.

To open a workspace window of a diagram, double-click the diagram item in the event tree. Using the window, you can edit the behavior. See “Behavior Overview” section about the behavior in more detail.



To delete a diagram, click the diagram item, then press the delete key. Click “Ok” button on the confirmation dialog if you sure want to delete it.



Use “New Diagram” button on the toolbar to add a new diagram into the current world. See following “Toolbar” section about the button in more detail.

## ***ToolBar***



### **New World**

Click to initialize the current world. After the initialization all nodes, routes diagrams, and modules are deleted. The current world became empty.



### **Load World**

Click to load a VRML2.0/97 file, and add the all node and route information into the current world.



### **Save World**

Click to save the current world into a VRML 97 file. When the saving starts, the behaviors are converted into some Script node and route information, and the class files are saved into the same directory.



### **Go Simulation**

Click to start the current simulation to start behavior actions. Note that you can't add any new events and diagrams, edit diagrams when the simulation is active. If you want to do the operations, you should stop the simulation.



### **Stop Simulation**

Click to stop the current simulation.



### **New Node**

Click to add a new node as a child node of a current selected node. Only node types that you can add into the selected node as the child are shown in the dialog. To add a new node as a top node, select "SceneGraph" item in SceneGraph tree.



### **New Route**

Click to add a new route. If the same route has been added already, this operation is ignored.



### **New Event**

Click to add a new user event. To create a new event, you have to select the event tab, and set or select the option values. If the same event has been added already, this operation is ignored. See “Behavior Overview” section about the user events in more detail.



### **New Diagram**

Click to add a new diagram. To create a new diagram, you have to select an event that is trigger to run this diagram, and set a name. If the same diagram has been added already, this operation is ignored. When the new diagram is created, the workspace window is created too.

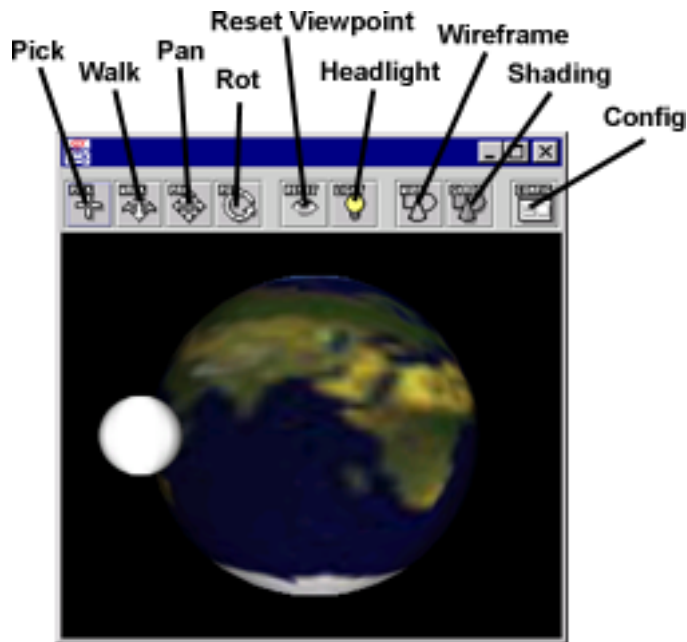


### **About**

Click to see CyberToolbox information.

## **Perspective Window**

This window shows the current virtual world using Java3D. When the simulation is active, the window is updated with the behaviors. Using a mouse, you can move in the world and pick objects.



## **Toolbar**



**Pick**

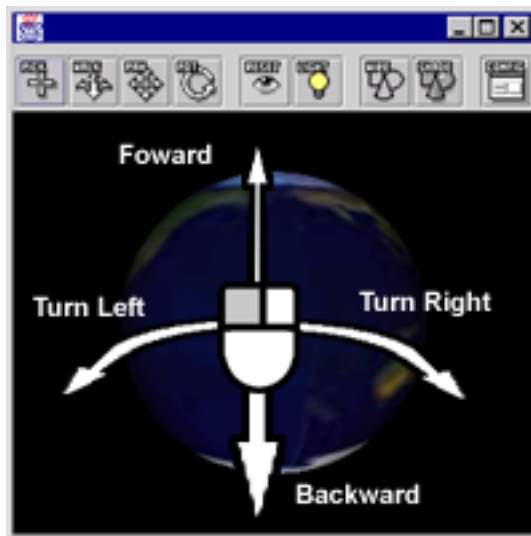
Click to switch into Pick mode. Pick mode allows you to pick objects in the window. When an object is clicked, the picking event happens. See “Behavior Overview” about the picking event in more detail.





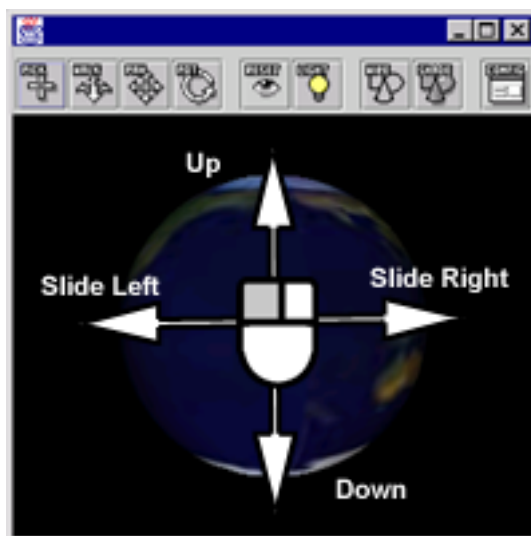
### Walk

Click to switch into Walk mode. Walk mode allows you to walk in the world. You can move forward when a mouse cursor position is in top half of the window, move backward when the position is in bottom of half, turn left when the position is in left half, turn right when the position is in right half.



### Pan

Click to switch into Pan mode. Pan mode allows you to translate a current viewpoint vertically or horizontally. You can up when a mouse cursor position is in top half of the window, down when the position is in bottom of half, slide left when the position is in left half, slide right when the position is in right half.

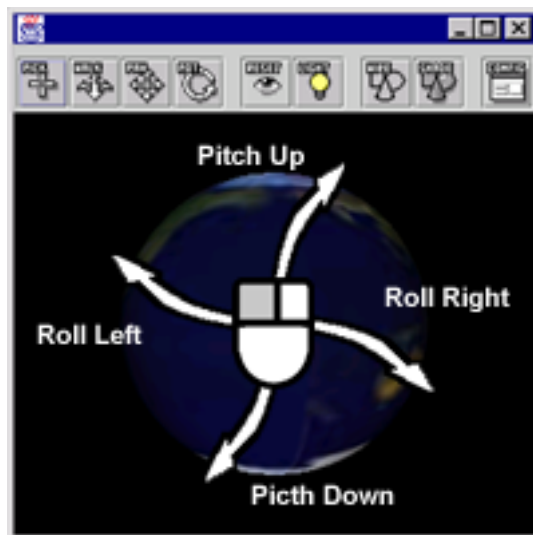






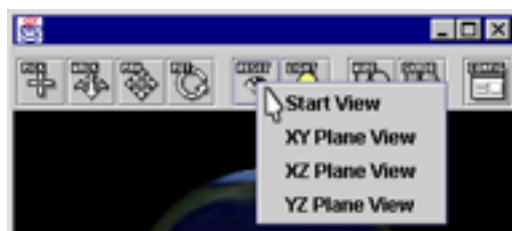
### Rot

Click to switch into Rot mode. Rot mode allows you rotate a current viewpoint. You can pitch up when a mouse cursor position is in top half of the window, pitch down when the position is in bottom of half, roll left when the position is in left half, roll right when the position is in right half.



### Reset Viewpoint

Click to move a current viewpoint into a position that you can see all objects in the world, and you can select the position in the pop-up menu.



### Headlight

Click to turns the headlight on and off. Try to turn on the headlight when your world has no lights.



### Wire-frame

Click to change the rendering style into the wire-frame mode. When the rendering style is the wire-frame mode, all objects are rendered as wire-frame entities.



### Shading

Click to change the rendering style into the normal mode. When the rendering style is the normal mode, all objects are rendered as graphical entities with the color or the texture.



### Config

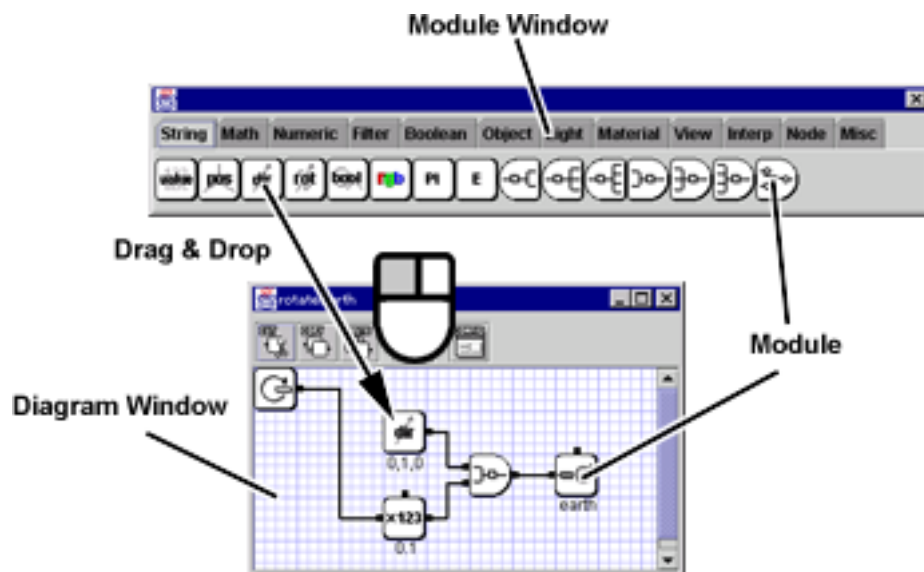
Click to set window properties, a rendering style, a navigation speed, a headlight state. The navigation speed is used a sensitivity value when you move in the world using a mouse.



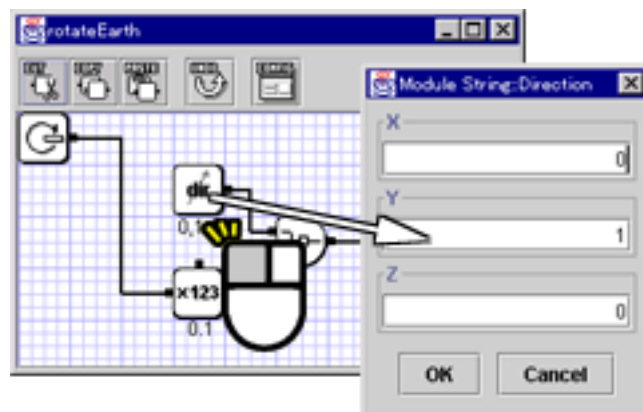
## **Diagram / Module Window**

The Diagram window is a workspace that you can create behaviors in a current world, and you can create the behaviors to connect between modules in the Module window

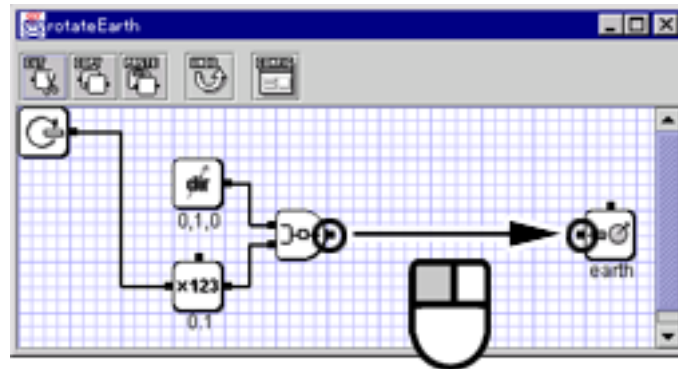
To add a new module into a diagram, drag the module in the Module window, and drop on the diagram window.



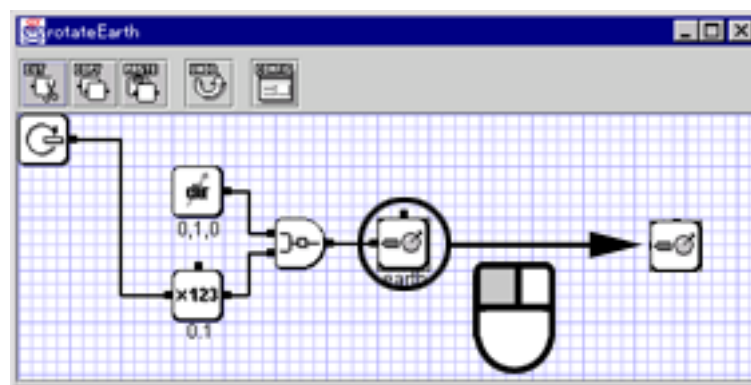
The module may have a setting dialog to set an inside value or a target node. To open the dialog, double-click the module.



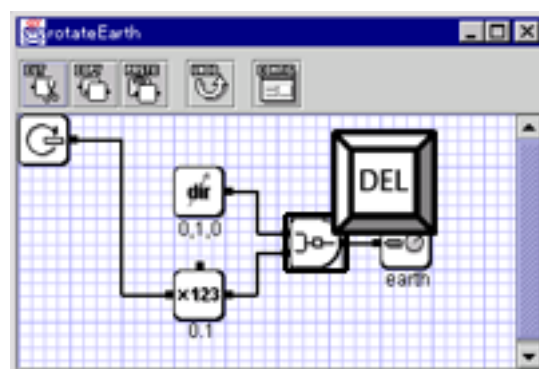
To add a data-flow line that connects between two module nodes, select the node, then drag the data-flow line, and drop on the other node.



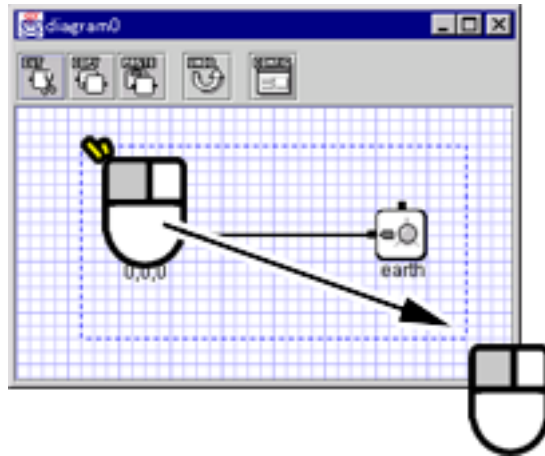
To move a module in a diagram window, drag the module.



To delete a module or a data-flow line, click the module or the node, then push DEL key.



To cut or copy modules and data-flow lines in a selecting box into a system clipboard. To select the box, click the start point, then drag to the end point.



## ***Toolbar***



**Cut**

Click to cut modules and data-flow lines in a current selecting box into a system clip board.



**Copy**

Click to copy modules and data-flow lines in a current selecting box into a system clip board.



**Paste**

Click to paste modules and data-flow in a system clip board into a current diagram.



**Undo**

Click to undo the latest operation.



**Config**

Click to set diagram properties, a name, a data-flow-line style.

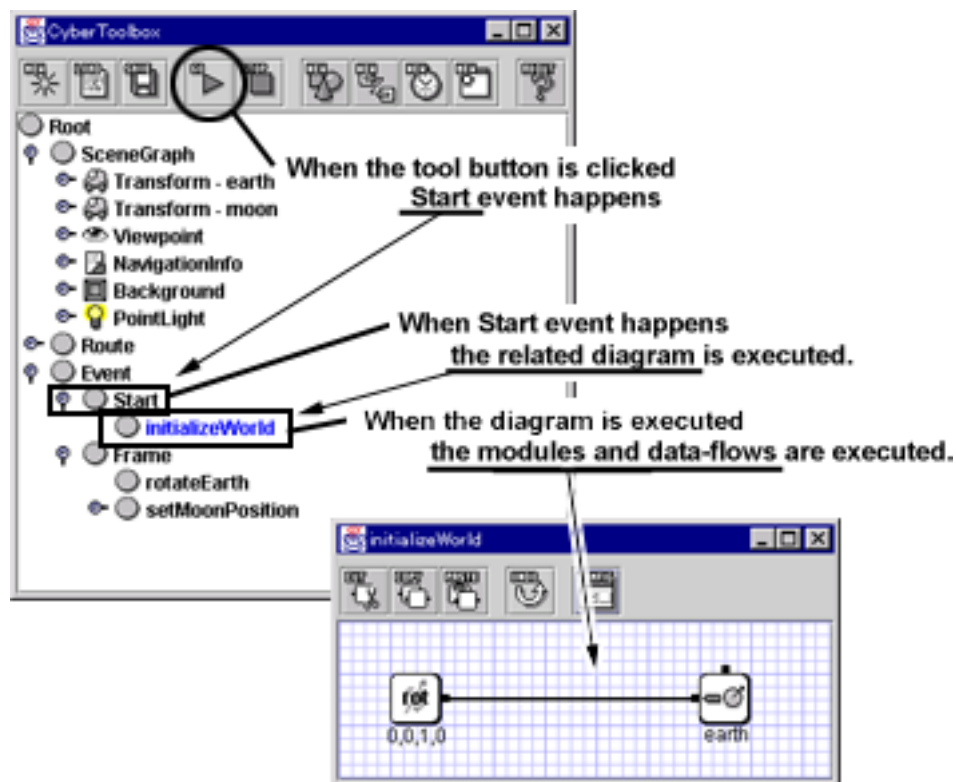
## Behavior Overview

CyberToolbox has a visual programming language for VRML97/2.0, everyone can create the behaviors using the language easily.

The behavior is executed by a trigger that happens in the current world, and the trigger is called as a event. For example, Start event happens when the simulation is started, Pick event happens when an object is clicked.

When an event happens, the related diagrams are executed. The diagram has modules and data-flows to define a behavior, and the modules and data-flows are executed.

For example, when Start event happens in the following world, a diagram that named “initializeWorld” is executed finally.



## **Event**

CyberToolbox for Java has only two system events and four user events in current release. The system events are added at first, you can't remove it. About the user events, you can create the new events with an option parameter, and remove it.

### ***System Event***

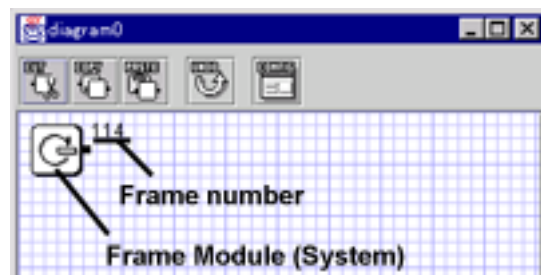
#### **Start**

This event happens at once when the simulation is started to click a tool button, “Go Simulation”, in World window. Use the event if you want to create behaviors for initialization purposes.



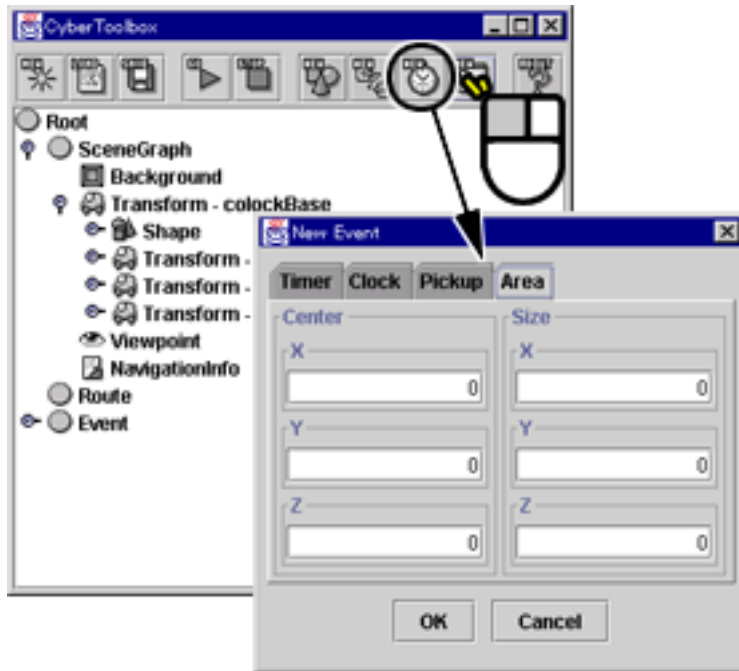
#### **Frame**

This event happens at ten times per second after the simulation is started. The related diagram has a system module as default, and the module output a current frame number.



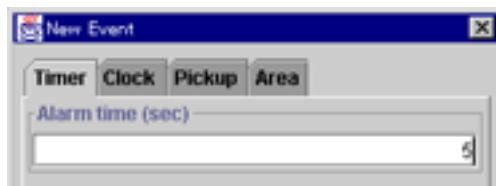
## User Event

CyberToolbox has four user event types that are added with the option values by user. To add the new event, click a tool button, “New Event”, in World window, then select the event tab, set the option value, and click OK.



### Timer

This event happens in a specified second after a simulation is started. To add the new event, set the alarm time.



### Clock

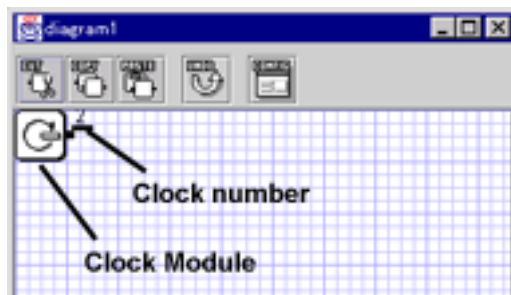
This event is similar to Frame event, and the event happens every a specified interval time. To add the new event, set the interval time.



The related diagram has a system module as default, and the module output a clock

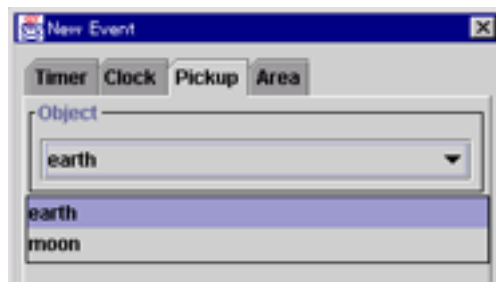


number.

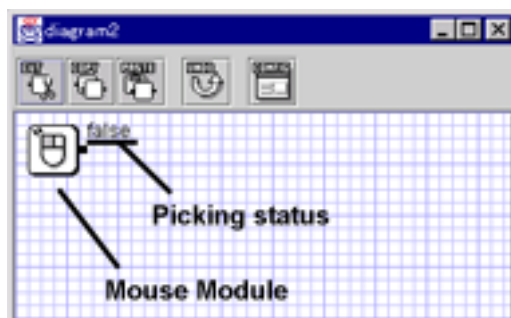


## Pickup

This event happens when a specified object is clicked in the perspective window. To add the new event, select an object in the list box. In the list box, only named Transform nodes are displayed as the objects.

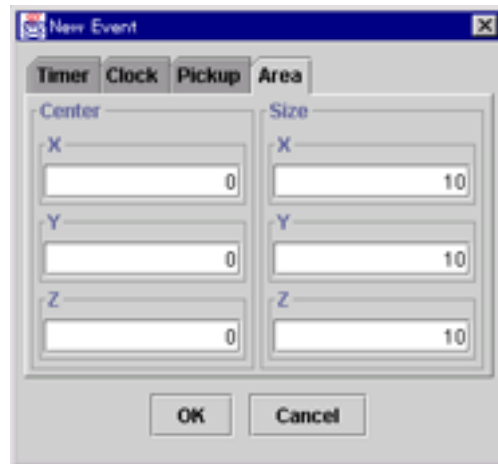


The related diagram has a system module as default, the module output a true string when you click the object, and output a false string when you release the object.

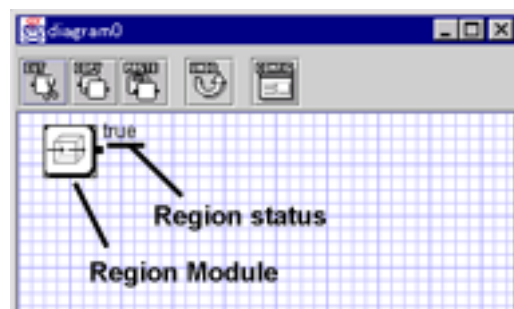


## Area

This event happens when you enter or exit in a specified region in the world. To add the new event, set the center and the size.

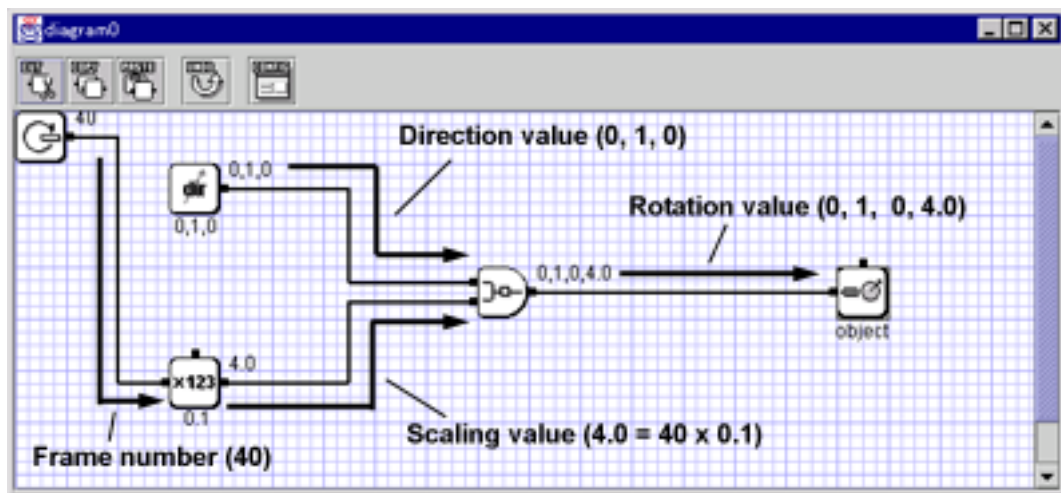


The related diagram has a system module as default, the module output a true string when you enter the region, and output a false value when you exit the region.



## **Diagram**

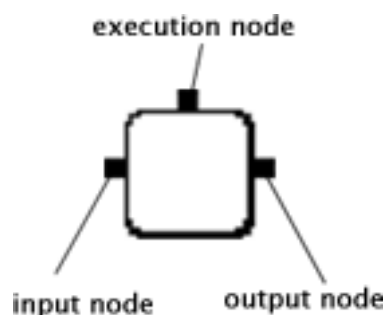
Diagram is a workspace that you can create behaviors using modules in the Module window. The connected line between the module nodes is a data-flow line, the module send string data from the output node to a input node of the other module.



The most top module in the data-flow is executed at first, the module sends string data from the output nodes to other modules that are connected the data-flow line with the output nodes, then the other modules sends string data to other modules similarly in data-flow sequence.

## **Module**

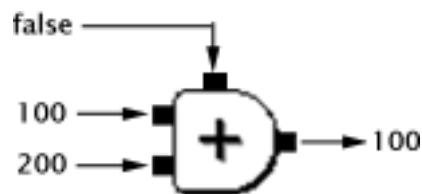
Module is a minimum unit to create behaviors, the module has three node types, a input node, a output node and a execution node.



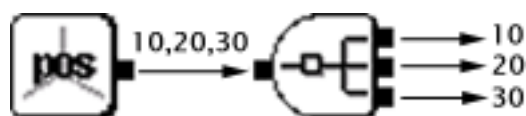
The input node type input a string data from an output node from other module, the output node type output a string data that is generated using the input data or the inside data. For example, a following module has two input nodes and an output node. The output result is a string data that is added two input string data.



Using the execution node type, you can set if the module is executed. If the execution node is not connected with a data-flow line from other module, the module isn't executed. When the execution node is connected, the module is executed when the input data is "true". When the module isn't executed when the input data is not "true". For example, a following module has two input nodes, an output node and a execution node which is received a "false" string. The output result is the same as the input string data because the module isn't executed.



The all node data format are string. When a module has to calculate the string data as a number, the module convert the string data into a number at first, then the module start the calculation. The string data can has some numbers to merge the numbers into a string using commas (','), you can merge some numbers into a string, or divide a string into some number strings. For example, a following left module output a position string which has three numbers, and the right module divide the string into three number string.



## Module Behavior Overview

---

Modules are classified into twelve classes, String, Numeric, Math, Filter, Boolean, Geom (Geometry), Object, Material, Light, View, Interp (Interpolator), Misc.

### **String**



#### **Value**

This module outputs a string that you set using the setting dialog.

Input node names	-
Output node names	OutValue
Execution node	-
Setting dialog	O
Target node	-
Result	OutValue = User setting value



#### **Position**

This module outputs a position string that you set using the setting dialog.

Input node names	-
Output node names	OutValue
Execution node	-
Setting dialog	O
Target node	-
Result	OutValue = User setting value (x, y, z)



#### **Direction**

This module outputs a direction string that you set using the setting dialog.

Input node names	-
Output node names	OutValue
Execution node	-
Setting dialog	O
Target node	-

Result	OutValue = User setting value (x, y, z)
--------	---



## Rotation

This module outputs a rotation string that you set using the setting dialog.

Input node names	-
Output node names	OutValue
Execution node	-
Setting dialog	O
Target node	-
Result	OutValue = User setting value (x, y, z, angle)



## Bool

This module outputs a boolean string that you set using the setting dialog.

Input node names	-
Output node names	OutValue
Execution node	-
Setting dialog	O
Target node	-
Result	OutValue = User setting value (true or false)



## Color

This module outputs a color string that you set using the setting dialog.

Input node names	-
Output node names	OutValue
Execution node	-
Setting dialog	O
Target node	-
Result	OutValue = User setting value (r, g, b)



**PI**

This module outputs a ratio of the circumference of a circle string that you set using the setting dialog.

Input node names	-
Output node names	OutValue
Execution node	-
Setting dialog	O
Target node	-
Result	OutValue = PI



**E**

This module outputs a base of the natural logarithms string that you set using the setting dialog.

Input node names	-
Output node names	OutValue
Execution node	-
Setting dialog	O
Target node	-
Result	OutValue = E



**Divide2Values**

This module divides an input string into two output strings.

Input node names	InValue (value1,value2)
Output node names	OutValue1 OutValue2
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue1 = value1 OutValue2 = value2
Example	InValue = 100,200 OutValue1 = 100 OutValue2 = 200



### Divide3Values

This module divides an input string into three output strings.

Input node names	InValue (value1,value2, value3)
Output node names	OutValue1 OutValue2 OutValue3
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue1 = value1 OutValue2 = value2 OutValue3 = value3
Example	InValue = 100,200,300 OutValue1 = 100 OutValue2 = 200 OutValue3 = 300



### Divide4Values

This module divides an input string into four output strings.

Input node names	InValue (value1,value2, value3,value4)
Output node names	OutValue1 OutValue2 OutValue3 OutValue4
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue1 = value1 OutValue2 = value2 OutValue3 = value3 OutValue4 = value4



Example	InValue = 100,200,300,400 OutValue1 = 100 OutValue2 = 200 OutValue3 = 300 OutValue4 = 400
---------	---



### Merge2Values

This module merges two input strings into an output string.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue = InValue1,InValue2
Example	InValue1 = 100 InValue2 = 200 OutValue1 = 100,200



### Merge3Values

This modules merges three input strings into an output string.

Input node names	InValue1 InValue2 InValue3
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue = InValue1,InValue2,InValue3
Example	InValue1 = 100 InValue2 = 200 InValue3 = 300 OutValue1 = 100,200,300



## Merge4Values

This module merges four input strings into an output string.

Input node names	InValue1 InValue2 InValue3 InValue4
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue = InValue1,InValue2,InValue3,InValue4
Example	InValue1 = 100 InValue2 = 200 InValue3 = 300 InValue4 = 400 OutValue1 = 100,200,300,400



## Selector

This module outputs either of the two input strings into the output node. When the execution node is not connected with a data-flow line or the execution node is received a “true” string, output the first input string into the output node. Otherwise, output the second input string.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-

Result	<pre> if (ExecutionNode is not connected)     OutValue = InValue1 else {     if (ExecutionNode data is "true")         OutValue = InValue1     else         OutValue = InValue2 } </pre>
Example	<pre> InValue1 = 100 InValue2 = 200 ExecutionNode = "false" OutValue = 200 </pre>

## **Numeric**

The all modules have an execution node. When the execution node is not connected with a data-flow line or the execution node is received a “true” string, output the calculation result into the output nodes. Otherwise, output the input strings as they are.



This module adds the two input values, and output the result into the output node.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = InValue1 + InValue2 else {     if (ExecutionNode data is "true")         OutValue = InValue1 + InValue2     Else         OutValue = InValue1 }</pre>
Example	<p>Example 1:</p> <p>InValue1 = 100 InValue2 = 200 OutValue = 300</p> <p>Example 2:</p> <p>InValue1 = 100,200,300 InValue2 = 400, 500,600 OutValue = 500,700,900</p>



## Minus

This module subtracts the second input value from the first input value, and output the result into the output node.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = InValue1 - InValue2 else {     if (ExecutionNode data is "true")         OutValue = InValue1 - InValue2     Else         OutValue = InValue1 }</pre>
Example	<p>Example 1:</p> <p>InValue1 = 200 InValue2 = 100 OutValue = 100</p> <p>Example 2:</p> <p>InValue1 = 600,700,800 InValue2 = 400, 500,600 OutValue = 200,200,200</p>



## Multi

This module multiplies the two input values, and output the result into the output node.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-

Result	<pre> if (ExecutionNode is not connected)     OutValue = InValue1 x InValue2 else {     if (ExecutionNode data is "true")         OutValue = InValue1 x InValue2     Else         OutValue = InValue1     } </pre>
Example	<p>Example 1:</p> <p>InValue1 = 20 InValue2 = 30 OutValue = 600</p> <p>Example 2:</p> <p>InValue1 = 100,200,300 (pos or vector) InValue2 = 2 OutValue = 200,400,600</p> <p>Example 3:</p> <p>InValue1 = 0,0,1 (vector) InValue2 = 0,1,0,1.57 (rotation) OutValue = 1,0,0</p>



## Divide

This module divides the first input value by the second input value, and output the result into the output node.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-

Result	<pre> if (ExecutionNode is not connected)     OutValue = InValue1 / InValue2 else {     if (ExecutionNode data is "true")         OutValue = InValue1 / InValue2     else         OutValue = InValue1     } </pre>
Example	<p>Example 1:</p> <p>InValue1 = 600 InValue2 = 30 OutValue = 20</p> <p>Example 2:</p> <p>InValue1 = 200,400,600 (pos or vector) InValue2 = 2 OutValue = 100,200,300</p>



### Mod

This module divides the first input value by the second input value, and output the quotient into the first output node, and output the remainder into the second output node.

Input node names	InValue1 InValue2
Output node names	OutValue1 OutValue2
Execution node	O
Setting dialog	-
Target node	-

Result	<pre> if (ExecutionNode is not connected) {     OutValue1 = (InValue1 – (InValue1 % InValue2)) /                                    InValue2      OutValue2 = InValue1 % InValue2 } else {     if (ExecutionNode data is “true”) {         OutValue1 = (InValue1 – (InValue1 % InValue2)) /                                    InValue2          OutValue2 = InValue1 % InValue2     }     else {         OutValue1 = InValue1         OutValue2 = InValue2     } } </pre>
Example	InValue1 = 10 InValue2 = 3 OutValue = 1



## And

This module executes a logical AND operation on the two input values, and output the result into the output node.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre> if (ExecutionNode is not connected)     OutValue = InValue1 &amp; InValue2 else {     if (ExecutionNode data is “true”)         OutValue = InValue1 &amp; InValue2     else         OutValue = InValue1 } </pre>



Example	InValue1 = 1 InValue2 = 2 OutValue = 0
---------	--



**Or**

This module executes a logical OR operation on the two input values, and output the result into the output node.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre> if (ExecutionNode is not connected)     OutValue = InValue1   InValue2 else {     if (ExecutionNode data is "true")         OutValue = InValue1   InValue2     else         OutValue = InValue1     } </pre>
Example	InValue1 = 1 InValue2 = 2 OutValue = 3



**Xor**

This module executes a logical XOR operation on the two input values, and output the result into the output node.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-

Result	<pre> if (ExecutionNode is not connected)     OutValue = InValue1 ^ InValue2 else {     if (ExecutionNode data is "true")         OutValue = InValue1 ^ InValue2     else         OutValue = InValue1     } </pre>
Example	<pre> InValue1 = 1 InValue2 = 2 OutValue = 3 </pre>

## **Math**

The all modules have an execution node. When the execution node is not connected with a data-flow line or the execution node is received a “true” string, output the calculation result into the output nodes. Otherwise, output the input strings as they are.



### **Increment**

This module adds 1 into the input value, and output the result into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = InValue + 1 else {     if (ExecutionNode data is "true")         OutValue = InValue + 1     else         OutValue = InValue }</pre>
Example	<pre>InValue = 1.1 OutValue = 2.1</pre>



### **Decrement**

This module subtracts 1 from the input value, and output the result into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-

Result	<pre> if (ExecutionNode is not connected)     OutValue = InValue - 1 else {     if (ExecutionNode data is "true")         OutValue = InValue - 1     else         OutValue = InValue     } </pre>
Example	InValue = 1 OutValue = 0



### Abs

This module outputs an absolute value of the input value into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre> if (ExecutionNode is not connected)     OutValue =   InValue   else {     if (ExecutionNode data is "true")         OutValue =   InValue       else         OutValue = InValue     } </pre>
Example	InValue = -1 OutValue = 1



## Negative

This module outputs a negative value of the input value into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = - InValue else {     if (ExecutionNode data is "true")         OutValue = - InValue     else         OutValue = InValue }</pre>
Example	<pre>InValue = 1 OutValue = -1</pre>



## Pow

This module outputs a value of the first input value raised to the power of the second input value into the output node.

Input node names	<pre>InValue1 InValue2</pre>
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = pow( InValue1, InValue2) else {     if (ExecutionNode data is "true")         OutValue = pow( InValue1, InValue2)     else         OutValue = InValue }</pre>

Example	InValue1 = 2 InValue2 = 3 OutValue = 8
---------	--



### Sqrt

This module outputs a square root value of the input value into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre> if (ExecutionNode is not connected)     OutValue = sqrt(InValue) else {     if (ExecutionNode data is "true")         OutValue = sqrt(InValue)     else         OutValue = InValue } </pre>
Example	InValue = 9 OutValue = 3



### Min

This module outputs the smaller of the two input values into the output node.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-

Result	<pre> if (ExecutionNode is not connected) {     if (InValue1 &lt; InValue2)         OutValue = InValue1     else         OutValue = InValue2     OutValue = sqrt(InValue) } else {     if (ExecutionNode data is "true") {         if (InValue1 &lt; InValue2)             OutValue = InValue1         else             OutValue = InValue2     }     else         OutValue = InValue } </pre>
Example	<pre> InValue1 = 100 InValue2 = 200 OutValue = 100 </pre>



## Max

This module outputs the greater of the two input values into the output node.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-

Result	<pre> if (ExecutionNode is not connected) {     if (InValue1 &gt; InValue2)         OutValue = InValue1     else         OutValue = InValue2     OutValue = sqrt(InValue) } else {     if (ExecutionNode data is "true") {         if (InValue1 &gt; InValue2)             OutValue = InValue1         else             OutValue = InValue2     }     else         OutValue = InValue } </pre>
Example	<pre> InValue1 = 100 InValue2 = 200 OutValue = 100 </pre>



## Log

This module outputs a natural logarithm of the input value into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre> if (ExecutionNode is not connected)     OutValue = log( InValue) else {     if (ExecutionNode data is "true")         OutValue = log( InValue)     else         OutValue = InValue } </pre>





## Exp

This module outputs a exponential number of raised to the power of the input value into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = exp( InValue) else {     if (ExecutionNode data is "true")         OutValue = exp( InValue)     else         OutValue = InValue }</pre>



## Sin

This module outputs a trigonometric sine of the input value into the output node.

Input node names	RadianAngle
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = sin( RadianAngle) else {     if (ExecutionNode data is "true")         OutValue = sin( RadianAngle)     else         OutValue = RadianAngle }</pre>



## Cos

This module outputs a trigonometric cosine of the input value into the output node.

Input node names	RadianAngle
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = cos( RadianAngle) else {     if (ExecutionNode data is "true")         OutValue = cos( RadianAngle)     else         OutValue = RadianAngle }</pre>



## Tan

This module outputs a trigonometric tangent of the input value into the output node.

Input node names	Radian
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = tan( RadianAngle) else {     if (ExecutionNode data is "true")         OutValue = tan( RadianAngle)     else         OutValue = RadianAngle }</pre>



## ASin

This module outputs an arc sin of the input value into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = asin( InValue) else {     if (ExecutionNode data is "true")         OutValue = asin( InValue)     else         OutValue = InValue }</pre>



## ACos

This module outputs an arc cosin of the input value into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = acos( InValue) else {     if (ExecutionNode data is "true")         OutValue = acos( InValue)     else         OutValue = InValue }</pre>



### ATan

This module outputs an arc tangent of the input value into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre> if (ExecutionNode is not connected)     OutValue = atan( InValue) else {     if (ExecutionNode data is "true")         OutValue = atan( InValue)     else         OutValue = InValue     } </pre>



### Degree2Radiun

This module converts the input value of an angle measured in degrees to the equivalent angle measured in radians, and outputs the result into the output node.

Input node names	DegreeAngle
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	$OutValue = DegreeAngle / 180 \times \pi$



### Radiun2Degree

This module converts the input value of an angle measured in radians to the equivalent angle measured in degrees, and outputs the result into the output node.

Input node names	DegreeAngle
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-

Result	OutValue = DegreeAngle / 180 x PI
--------	-----------------------------------

## **Filter**

The all modules have an execution node. When the execution node is not connected with a data-flow line or the execution node is received a “true” string, output the calculation result into the output nodes. Otherwise, output the input strings as they are.



### **Scale**

This module scales the input value using the setting scaling value, and outputs the result into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	O
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = InValue * User setting value else {     if (ExecutionNode data is "true")         OutValue = InValue * User setting value     else         OutValue = InValue }</pre>
Example	<p>InValue = 10 User setting value = 20 OutValue = 200</p>



### **Offset**

This module adds the setting offset value into the input value, and outputs the result into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	O
Target node	-

Result	<pre> if (ExecutionNode is not connected)     OutValue = InValue + User setting value else {     if (ExecutionNode data is "true")         OutValue = InValue + User setting value     else         OutValue = InValue     } </pre>
Example	InValue = 10 User setting value = 1000 OutValue = 1010



### Ceil

This module outputs the smallest value that is not less than the input value and is equal to a mathematical integer into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre> if (ExecutionNode is not connected)     OutValue = ceil(InValue) else {     if (ExecutionNode data is "true")         OutValue = ceil(InValue)     else         OutValue = InValue     } </pre>
Example	InValue = 12.3 OutValue = 13



## Floor

This module outputs the largest value that is not greater than the argument and is equal to a mathematical integer into the output node.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre>if (ExecutionNode is not connected)     OutValue = floor(InValue) else {     if (ExecutionNode data is "true")         OutValue = floor(InValue)     else         OutValue = InValue }</pre>
Example	<p>InValue = 12.3 OutValue = 12</p>



## High

This module outputs the setting high value when the input value is greater than the setting high value into the output node. Otherwise, the module outputs the input value as it is.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	O
Target node	-



Result	<pre> if (ExecutionNode is not connected) {   if (User setting high value &lt; InValue)     OutValue = User setting high value   else     OutValue = InValue } else {   if (ExecutionNode data is "true") {     if (User setting hi value &lt; InValue)       OutValue = User setting high value     else       OutValue = InValue   }   else     OutValue = InValue } </pre>
Example	InValue = 120 User setting high value = 100 OutValue = 100



### Low

This module outputs the setting low value when the input value is less than the setting low value into the output node. Otherwise, the module outputs the input value as it is.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	O
Target node	-

Result	<pre> if (ExecutionNode is not connected) {   if (InValue &lt; User setting low value)     OutValue = User setting low value   else     OutValue = InValue } else {   if (ExecutionNode data is "true") {     if (InValue M ¥¥&lt; User setting low data)       OutValue = User setting log value     else       OutValue = InValue   }   else     OutValue = InValue } </pre>
Example	InValue = 12.3 OutValue = 12



## Ragne

This module outputs the setting high value when the input value is greater than the setting high value into the output node, or outputs the setting low value when the input value is less than the setting low value into the output node. Otherwise, the module outputs the input value as it is.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	O
Target node	-

Result	<pre> if (ExecutionNode is not connected) {   if (InValue &lt; User setting low value)     OutValue = User setting low value   else     OutValue = InValue   if (User setting high value &lt; InValue)     OutValue = User setting high value   else     OutValue = InValue } else {   if (ExecutionNode data is "true") {     if (InValue M ¥¥&lt; User setting low data)       OutValue = User setting log value     else       OutValue = InValue     if (User setting high value &lt; InValue)       OutValue = User setting high value     else       OutValue = InValue   }   else     OutValue = InValue } </pre>
Example	<pre> InValue = 12.3 OutValue = 12 </pre>

## **Boolean**



### **Equal**

This module outputs a “true” string into the output node when the first input value is equal with the second input value. Otherwise, the module outputs a “false” string.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	if (InValue1 == InValue2) OutValue = “true” else OutValue = “false”



### **NotEqual**

This module outputs a “true” string into the output node when the first input value is not equal with the second input value. Otherwise, the module outputs a “false” string.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	if (InValue1 != InValue2) OutValue = “true” else OutValue = “false”



## Greater

This module outputs a “true” string into the output node when the first input value is greater than the second input value. Otherwise, the module outputs a “false” string.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	if (InValue1 > InValue2) OutValue = “true” else OutValue = “false”



## Less

This module outputs a “true” string into the output node when the first input value is less than the second input value. Otherwise, the module outputs a “false” string.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	if (InValue1 < InValue2) OutValue = “true” else OutValue = “false”



### Equal Greater

This module outputs a “true” string into the output node when the first input value is equal with the second input value or greater than the second input value. Otherwise, the module outputs a “false” string.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	if (InValue1 >= InValue2) OutValue = “true” else OutValue = “false”



### Equal Less

This module outputs a “true” string into the output node when the first input value is equal with the second input value or less than the second input value. Otherwise, the module outputs a “false” string.

Input node names	InValue1 InValue2
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	if (InValue1 <= InValue2) OutValue = “true” else OutValue = “false”



### Not

This module outputs a “false” string into the output node when the input value is equal with a “true” string. Otherwise, the module outputs a “true” string.

Input node names	InValue
Output node names	OutValue
Execution node	O
Setting dialog	-
Target node	-
Result	<pre> if (ExecutionNode is not connected)     OutValue = ! InValue else {     if (ExecutionNode data is “true”) {         OutValue = ! InValue     }     else         OutValue = InValue     } </pre>



### And

This module outputs a “true” string into the output node when the first input value and the second input value are “true”. Otherwise, the module outputs a “false” string.

Input node names	InValue
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue = ! InValue



### Or

This module outputs a “true” string into the output node when the first input value or the second input value is “true”. Otherwise, the module outputs a “false” string.

Input node names	InValue
Output node names	OutValue
Execution node	-
Setting dialog	-

Target node	-
Result	OutValue = ! InValue



## Geom (Geometry)



### Normalize

This module normalizes an input 3D vector, then outputs the normalized vector into the output node.

Input node names	InValue (x, y, z)
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue = normalize(InValue)



### Inverse

This module negates an input 3D vector, then outputs the negated vector into the output node.

Input node names	InValue (x, y, z)
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue = -x, -y, -z



### GetLength

This module get a length of an input 3D vector, then outputs the length into the output node.

Input node names	InValue (x, y, z)
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue = sqrt(x*x + y*y + z*z)



### GetDot

This module get a inner product of two input 3D vectors, then outputs the value into the output node.

Input node names	InValue (x1, y1, z1) InValue (x2, y2, z2)
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	$OutValue = (x1 \cdot x2) + (y1 \cdot y2) + (z1 \cdot z2)$



### GetAngle

This module get a radian angle between two input 3D vectors, then outputs the angle into the output node.

Input node names	InValue (x1, y1, z1) InValue (x2, y2, z2)
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	$OutValue = \arccos( ((x1 \cdot x2) + (y1 \cdot y2) + (z1 \cdot z2)) / (\sqrt{x1^2 + y1^2 + z1^2} + \sqrt{x2^2 + y2^2 + z2^2}) )$



### GetVector

This module get a vector from the first input value to the second input value, then outputs the vector into the output node.

Input node names	InValue (x1, y1, z1) InValue (x2, y2, z2)
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	$OutValue = (x2 - x1), (y2 - y1), (z2 - z1)$



### GetDistance

This module get a distance between two input 3D vectors, then outputs the distance into the output node.

Input node names	InValue (x1, y1, z1) InValue (x2, y2, z2)
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	$\text{OutValue} = \sqrt{(x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2}$



### GetCross

This module get a cross vector of two input 3D vectors, then outputs the vector into the output node.

Input node names	InValue (x1, y1, z1) InValue (x2, y2, z2)
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue = corss vector



### Rotate

This module rotate the first input 3D vector using the second Rotation value, then outputs the rotated vector into the output node.

Input node names	InValue1 (x, y, z) InValue2 (x, y, z, angle)
Output node names	OutValue
Execution node	-
Setting dialog	-
Target node	-
Result	OutValue = rotated vector

## **Object**



### **SetLocation**

This module sets the location field of the Transform node to the position defined by the input value.

Input node names	Location (x, y, z)
Output node names	-
Execution node	O
Setting dialog	O
Target node	Transform
Result	Transform : : location = location



### **SetRotation**

This module sets the rotation field of the Transform node to the rotation defined by the input value.

Input node names	Rotation (x, y, z, angle)
Output node names	-
Execution node	O
Setting dialog	O
Target node	Transform
Result	Transform : : rotation = rotation



### **SetScale**

This module sets the scale field of the Transform node to the scale defined by the input value.

Input node names	Scale (x, y, z)
Output node names	-
Execution node	O
Setting dialog	O
Target node	Transform
Result	Transform : : scale = Scale



### SetCenter

This module sets the center field of the Transform node to the center defined by the input value.

Input node names	Center (x, y, z)
Output node names	-
Execution node	O
Setting dialog	O
Target node	Transform
Result	Transform : : center = center



### GetLocation

This module outputs a current location of the location field of the Transform node into the output node.

Input node names	-
Output node names	Location (x, y, z)
Execution node	-
Setting dialog	O
Target node	Transform
Result	Location = Transform : : location



### GetRotation7

This module outputs a current rotation of the rotation field of the Transform node into the output node.

Input node names	-
Output node names	Rotation (x, y, z, angle)
Execution node	-
Setting dialog	O
Target node	Transform
Result	Rotation = Transform : : rotation



### GetScale

This module outputs a current scale of the scale field of the Transform node into the output node.

Input node names	-
Output node names	Scale (x, y, z)
Execution node	-
Setting dialog	O
Target node	Transform
Result	Scale = Transform : : scale



### GetCenter

This module outputs a current center of the center field of the Transform node into the output node.

Input node names	-
Output node names	Center (x, y, z)
Execution node	-
Setting dialog	O
Target node	Transform
Result	Center= Transform : : center

## **Material**



### **SetAmbientIntensity**

This module sets the ambientIntensity field of the Material node to the color defined by the input value.

Input node names	AmbientIntensity
Output node names	-
Execution node	O
Setting dialog	O
Target node	Material
Result	Material : : ambientIntensity = AmbientIntensity



### **SetDiffuseColor**

This module sets the diffuseColor field of the Material node to the color defined by the input value.

Input node names	DiffuseColor (r, g, b)
Output node names	-
Execution node	O
Setting dialog	O
Target node	Material
Result	Material : : diffuseColor = DiffuseColor



### **SetEmissiveColor**

This module sets the emissiveColor field of the Material node to the color defined by the input value.

Input node names	EmissiveColor (r, g, b)
Output node names	-
Execution node	O
Setting dialog	O
Target node	Material
Result	Material : : emissiveColor = EmissiveColor



### SetSpecularColor

This module sets the specularColor field of the Material node to the color defined by the input value.

Input node names	SpecularColor (r, g, b)
Output node names	-
Execution node	O
Setting dialog	O
Target node	Material
Result	Material : : specularColor = SpecularColor



### SetShininess

This module sets the shininess field of the Material node to the shininess defined by the input value.

Input node names	Shininess
Output node names	-
Execution node	O
Setting dialog	O
Target node	Material
Result	Material : : shininess = Shininess



### GetAmbientIntensity

This module outputs a current color of the ambientIntensity field of the Material node into the output node.

Input node names	-
Output node names	AmbientIntensity
Execution node	-
Setting dialog	O
Target node	Material
Result	AmbientIntensity = Material : : ambientIntensity





### GetDiffuseColor

This module outputs a current color of the diffuseColor field of the Material node into the output node.

Input node names	-
Output node names	DiffuseColor (r, g, b)
Execution node	-
Setting dialog	O
Target node	Material
Result	DiffuseColor = Material : : diffuseColor



### GetEmissiveColor

This module outputs a current color of the emissiveColor field of the Material node into the output node.

Input node names	-
Output node names	EmissiveColor (r, g, b)
Execution node	-
Setting dialog	O
Target node	Material
Result	EmissiveColor = Material : : emissiveColor



### GetSpecularColor

This module outputs a current color of the specularColor field of the Material node into the output node.

Input node names	-
Output node names	SpecularColor (r, g, b)
Execution node	-
Setting dialog	O
Target node	Material
Result	SpecularColor = Material : : specularColor



### **GetShininess**

This module outputs a current value of the shininess field of the Material node into the output node.

Input node names	-
Output node names	Shininess
Execution node	-
Setting dialog	O
Target node	Material
Result	Shininess= Material : : shininess

## **Light**



### **SetOn**

This module sets the on field of the DirectionalLight, PointLight, or SpotLight node to the flag defined by the input value.

Input node names	On ("true" or "false")
Output node names	-
Execution node	O
Setting dialog	O
Target node	DirectionalLight / PointLight / SpotLight
Result	Light : : on = On



### **SetColor**

This module sets the color field of the DirectionalLight, PointLight, or SpotLight node to the color defined by the input value.

Input node names	Color (r, g, b)
Output node names	-
Execution node	O
Setting dialog	O
Target node	DirectionalLight / PointLight / SpotLight
Result	Light : : color = Color



### **SetIntensity**

This module sets the intensity field of the DirectionalLight, PointLight, or SpotLight node to the intensity defined by the input value.

Input node names	Intensity
Output node names	-
Execution node	O
Setting dialog	O
Target node	DirectionalLight / PointLight / SpotLight
Result	Light : : intensity = Intensity



### SetLocation

This module sets the location field of the PointLight or SpotLight node to the location defined by the input value.

Input node names	Location (x, y, z)
Output node names	-
Execution node	O
Setting dialog	O
Target node	PointLight / SpotLight
Result	Light : : location= Location



### SetDirection

This module sets the direction field of the DirectionalLight or SpotLight node to the direction defined by the input value.

Input node names	Direction (x, y, z)
Output node names	-
Execution node	O
Setting dialog	O
Target node	DirectionalLight / SpotLight
Result	Light : : direction = Direction



### SetRadius

This module sets the radius field of the PointLight or SpotLight node to the radius defined by the input value.

Input node names	Radius
Output node names	-
Execution node	O
Setting dialog	O
Target node	PointLight / SpotLight
Result	Light : : radius = Radius



### GetOn

This module outputs a current status of the on field of the DirectionalLight, PointLight, or SpotLight node into the output node.

Input node names	-
Output node names	On ("true" or "false")
Execution node	-
Setting dialog	O
Target node	DirectionalLight / PointLight / SpotLight
Result	On = Light : : on



### GetColor

This module outputs a current color of the color field of the DirectionalLight, PointLight, or SpotLight node into the output node.

Input node names	-
Output node names	Color (r, g, b)
Execution node	-
Setting dialog	O
Target node	DirectionalLight / PointLight / SpotLight
Result	Color = Light : : color



### GetIntensity

This module outputs a current intensity of the intensity field of the DirectionalLight, PointLight, or SpotLight node into the output node.

Input node names	-
Output node names	Intensity
Execution node	-
Setting dialog	O
Target node	DirectionalLight / PointLight / SpotLight
Result	Intensity = Light : : intensity



### GetLocation

This module outputs a current location of the location field of the PointLight or SpotLight node into the output node.

Input node names	-
Output node names	Location (x, y, z)
Execution node	-
Setting dialog	O
Target node	PointLight / SpotLight
Result	Location = Light : : location



### GetDirection

This module outputs a current direction of the direction field of the DirectionalLight or SpotLight node into the output node.

Input node names	-
Output node names	Direction (x, y, z)
Execution node	-
Setting dialog	O
Target node	DirectionalLight / SpotLight
Result	Intensity = Light : : intensity



### GetRadius

This module outputs a current radius of the radius field of the PointLight or SpotLight node into the output node.

Input node names	-
Output node names	Radius
Execution node	-
Setting dialog	O
Target node	PointLight / SpotLight
Result	Radius = Light : : radius

## **Viewpoint**



### **SetPosition**

This module sets the position field of the Viewpoint node to the position defined by the input value.

Input node names	Position (x, y, z)
Output node names	-
Execution node	O
Setting dialog	O
Target node	Viewpoint
Result	Viewpoint : : position = Position



### **SetOrientation**

This module sets the orientation field of the Viewpoint node to the orientation defined by the input value.

Input node names	Orientation (x, y, z, angle)
Output node names	-
Execution node	O
Setting dialog	O
Target node	Viewpoint
Result	Viewpoint : : orientaton = Orientation



### **SetFOV**

This module sets the fieldOfView field of the Viewpoint node to the FOV defined by the input value.

Input node names	Fov
Output node names	-
Execution node	O
Setting dialog	O
Target node	Viewpoint
Result	Viewpoint : : fieldOfView = fov



### GetPosition

This module outputs a current position of the position field of the Viewpoint node into the output node.

Input node names	-
Output node names	Position (x, y, z)
Execution node	-
Setting dialog	O
Target node	Viewpoint
Result	Position = Viewpoint : : position



### GetOrientation

This module outputs a current orientation of the orientation field of the Viewpoint node into the output node.

Input node names	-
Output node names	Orientation (x, y, z, angle)
Execution node	-
Setting dialog	O
Target node	Viewpoint
Result	Orientation = Viewpoint : : orientaton



### GetFOV

This module outputs a current FOV of the fieldOfView field of the Viewpoint node into the output node.

Input node names	
Output node names	fov
Execution node	-
Setting dialog	O
Target node	Viewpoint
Result	fov = Viewpoint : : fieldOfView



## **Interp (Interpolator)**



### **SetScalarFraction**

This module sets the set\_fraction field of the ScalarInterpolator node to the value defined by the input value.

Input node names	InValue (0.0 – 1.0)
Output node names	-
Execution node	O
Setting dialog	O
Target node	ScalarInterpolator
Result	ScalarInterpolator : : set_fraction = InValue



### **SetPositionFraction**

This module sets the set\_fraction field of the PositionInterpolator node to the value defined by the input value.

Input node names	InValue (0.0 – 1.0)
Output node names	-
Execution node	O
Setting dialog	O
Target node	PositionInterpolator
Result	PositionInterpolator: : set_fraction = InValue



### **SetNormalFraction**

This module sets the set\_fraction field of the NormalInterpolator node to the value defined by the input value.

Input node names	InValue (0.0 – 1.0)
Output node names	-
Execution node	O
Setting dialog	O
Target node	NormalInterpolator
Result	NormalInterpolator: : set_fraction = InValue



### SetOrientationFraction

This module sets the set\_fraction field of the OrientationInterpolator node to the value defined by the input value.

Input node names	InValue (0.0 – 1.0)
Output node names	-
Execution node	O
Setting dialog	O
Target node	OrientationInterpolator
Result	OrientationInterpolator: : set_fraction = InValue



### SetColorFraction

This module sets the set\_fraction field of the ColorInterpolator node to the value defined by the input value.

Input node names	InValue (0.0 – 1.0)
Output node names	-
Execution node	O
Setting dialog	O
Target node	ColorInterpolator
Result	ColorInterpolator : : set_fraction = InValue

## **Misc**



### **SetSwitch**

This module sets the whichChoice field of the Switch node to the value defined by the input value.

Input node names	InValue (0.0 – 1.0)
Output node names	-
Execution node	O
Setting dialog	O
Target node	SwitchNode
Result	SwitchNode : : whichChoice = InValue



### **SetSkyColor**

This module sets the first skyColor field of the Background node to the value defined by the input value.

Input node names	InValue (0.0 – 1.0)
Output node names	-
Execution node	O
Setting dialog	O
Target node	Background
Result	Background : : skyColor = InValue



### **GetTime**

This module outputs a current hour, minute and second into the output nodes.

Input node names	-
Output node names	Hour Minute Second
Execution node	-
Setting dialog	-
Target node	-

Result	Hour = current system hour Minute = current system minute Second = current system second
--------	--



## Random

This module outputs a random number greater than or equal to 0.0 and less than 1.0 into the output node. When the execution node is connected, the module change the random number only when the node received a “true” string.

Input node names	-
Output node names	RandomValue
Execution node	-
Setting dialog	-
Target node	-
Result	RandomValue = 0.0 – 1.0



## Beep

This module emits a beep sound when the execution node is received a “true” string.

Input node names	-
Output node names	-
Execution node	-
Setting dialog	-
Target node	-
Result	Play a beep sound



## JavaConsole

This module outputs the input string into the standard output stream.

Input node names	String
Output node names	-
Execution node	-
Setting dialog	-
Target node	-
Result	Output the String into Java Console