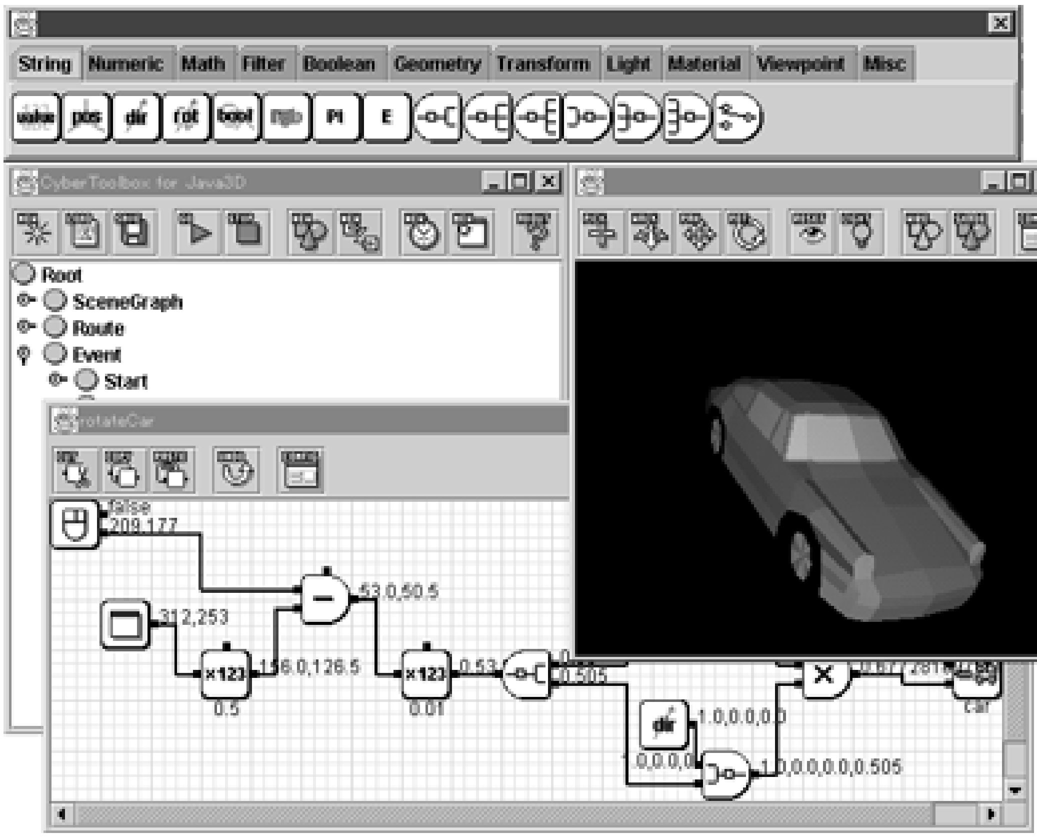


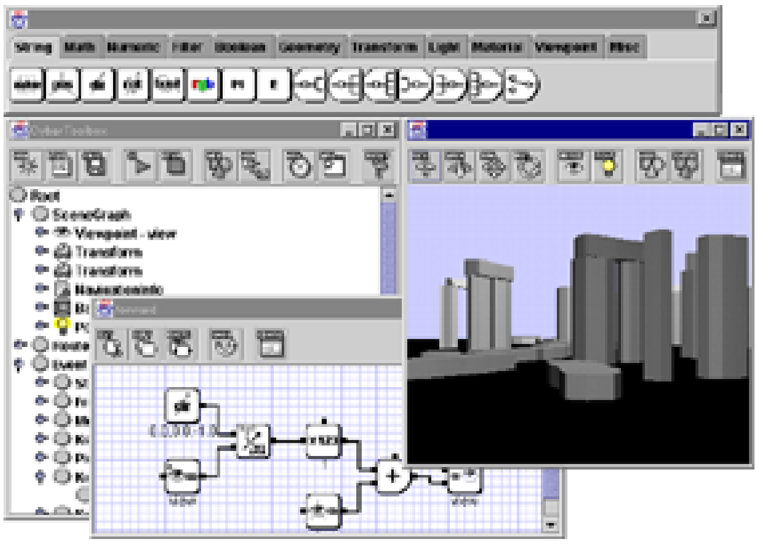
# CyberToolbox for Java3D

Release 1.2

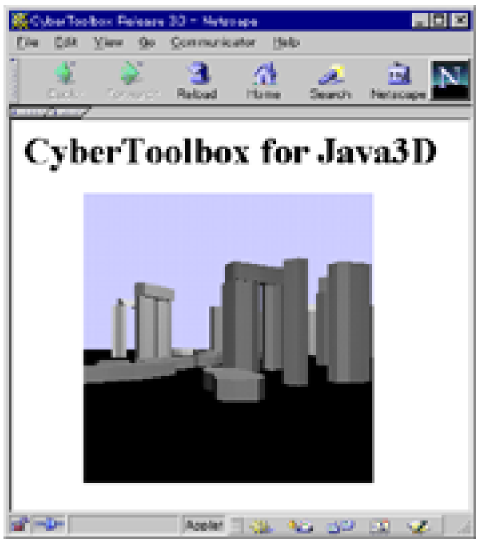
# User's Guide

## *What is CyberToolbox for Java3D ?*

CyberToolbox for Java3D *1) is an authoring tool of Virtual Reality on Java2 and Java3D platforms. The CyberToolbox is based on a scene graph of VRML97, however has a visual programming language with an original event model that is different from VRML97 to create more good behaviors easily.



The CyberToolbox is released with an applet package to browse contents that are created by the CyberToolbox using web browsers. Using the applet, everyone can browse the contents with Microsoft Internet Explore or Netscape Communicator on the Internet.



[1] Note : CyberToolbox for Java3D is a different product from CyberToolbox for Java that is based on VRML97 all.

I am developing CyberToolbox with CyberVRML97 that is a development package for VRML and Java3D. If you have any interest in VRML or Java3D application developments, yon can get the information in more detail from http://www.cyber.koganei.tokyo.jp.

## Installation

To use CyberToolbox for Java3D, you have to install latest Java2 (JDK1.2) and Java3D packages. If you want to use the sensor modules of virtual reality devices, you have to install latest Java Communications API, too. You can 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 classes in the VRML-Java3D package conflicts with the VRML-CyberToolbox package.

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

If you are a registration user of the CyberToolbox and you want to add your modules into the CyberToolbox, you have to copy a jar package, "tools.jar", into your JDK or JRE directory. The package is distributed with JDK.

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

```
jar xvf ctbj3d110.zip
```

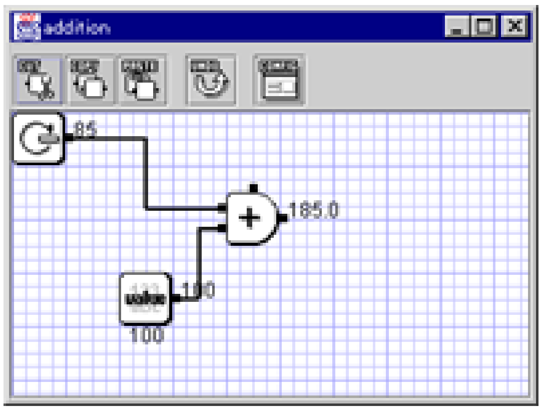
To run the CyberToolbox, use following commands.

```
cd cybertoolbox
run (or run.bat)
```

# Tutorial

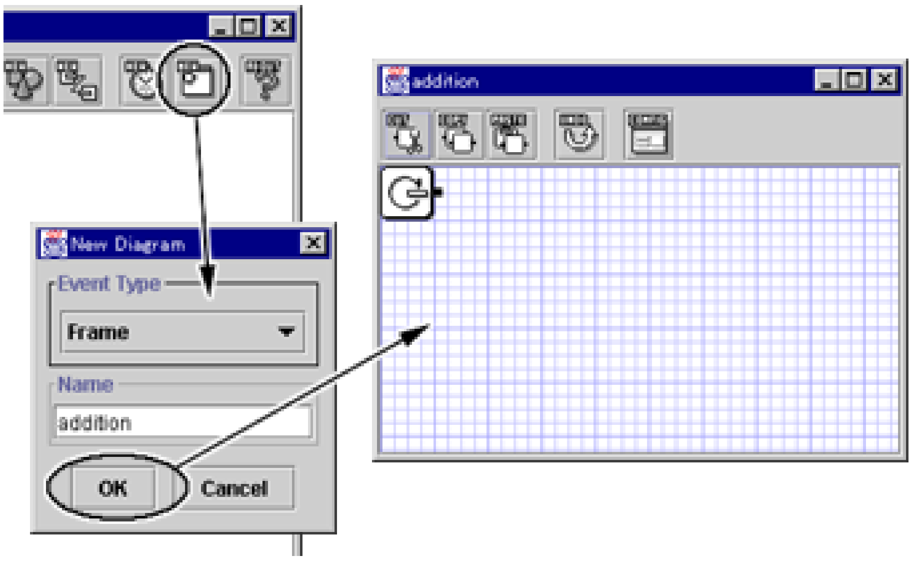
## Addition

This tutorial shows you how to create a simple behavior that adds two values.

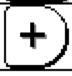


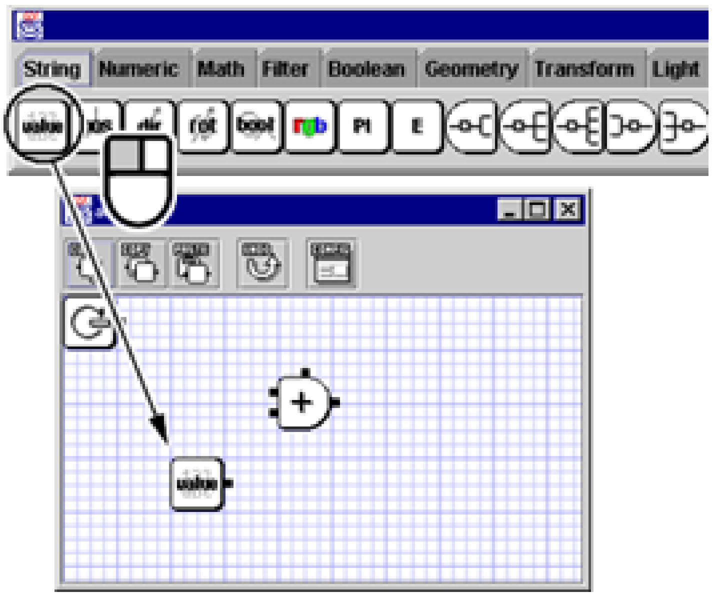
## 1. Creating a diagram to add two values

Create a diagram that is executed by Frame event. This event happens at ten times per second after the simulation is started. The diagram has a system module that outputs a current frame number. To create the diagram, push "New Diagram" button in World window, select "Frame" as the event type, and set "addition" to the name.



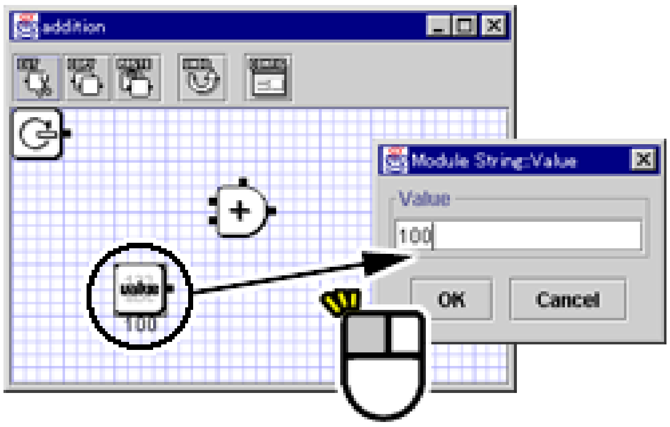
Drop the following two modules into the diagram from Module window,

| | Icon | Class | Name |
|---|---|---|---|
| 1 | | String | Value |
| 2 |  | Numeric | Add |

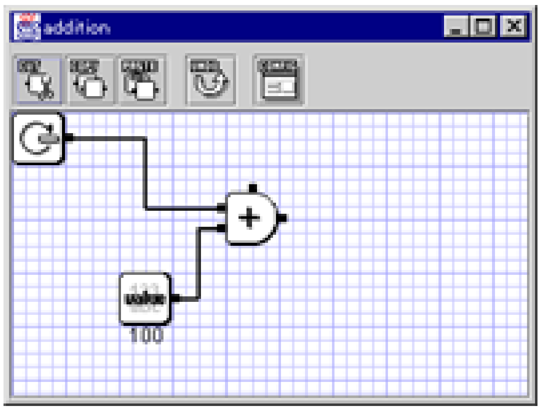


Double-click a following module, and set the values at "100".

| | Icon | Specified value |
|---|---|---|
| 1 | | 100 |

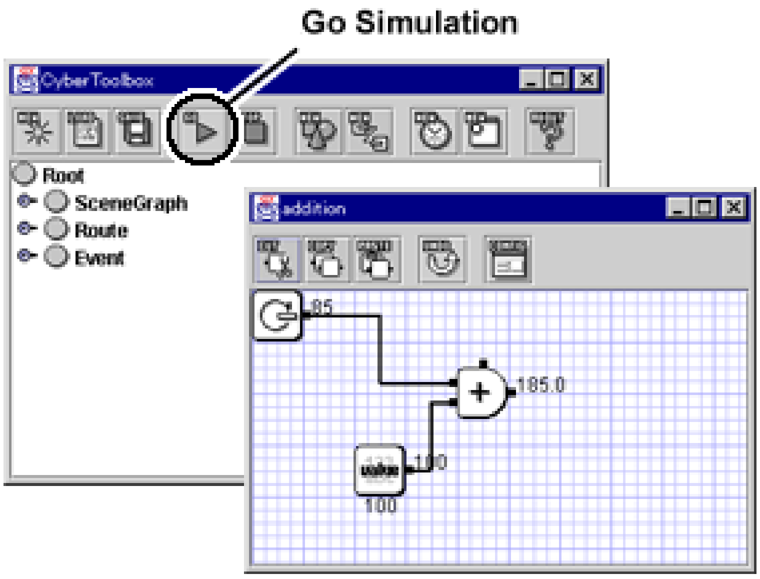


Connect the following data-flow lines.



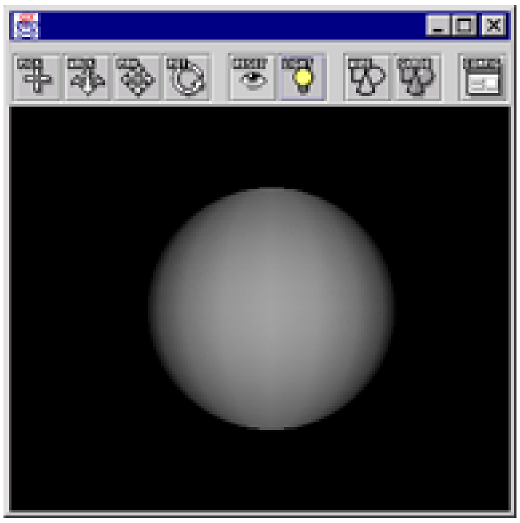
## 2. Starting the simulation

To run the behavior, click "Go Simulation" button in World window. When the simulation is active, the addition module adds 100 to a current frame number, and output the value into the output node.



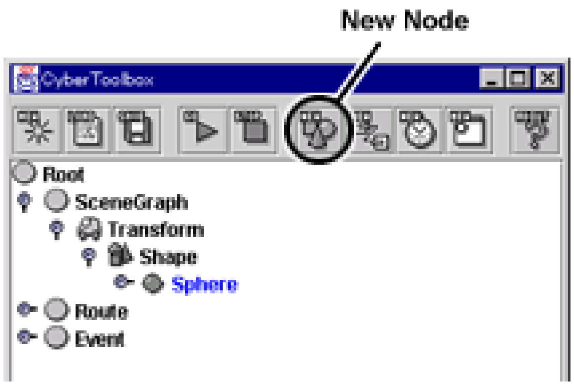
## Rotating ball

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



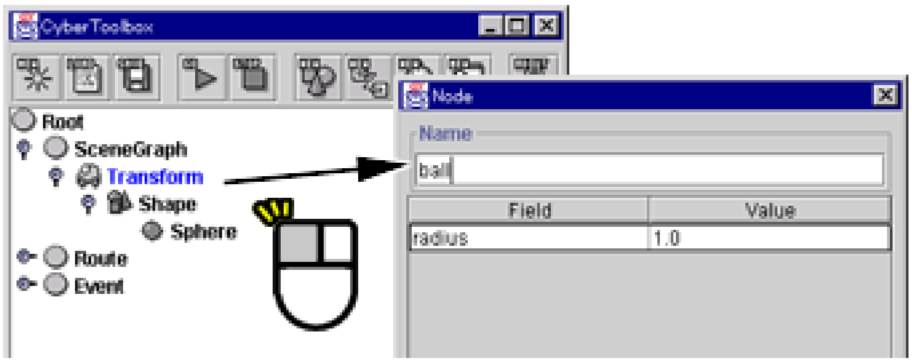
### 1. Creating a ball object

To add a ball object, you have to add three nodes into a current scene-graph using a “New Node” button in World window . 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.

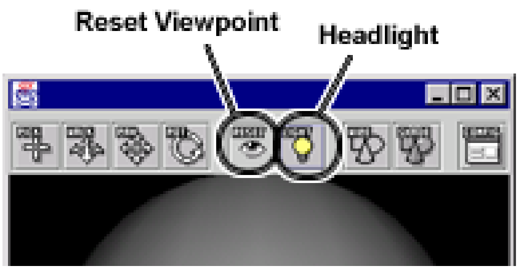


See “Operation Overview” about creating a new node in more detail.

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

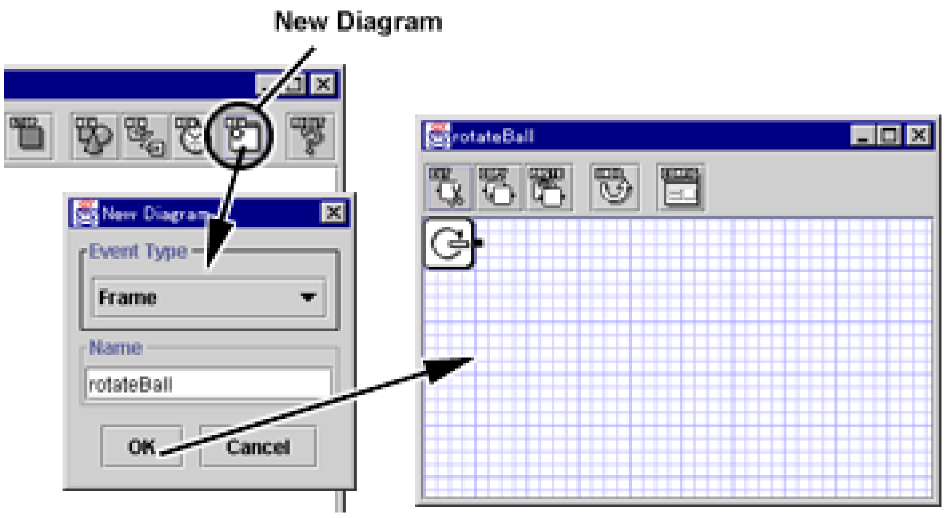


To see the added object in Perspective window, push "Reset Viewpoint " button and select "XY Plane View", then push "Headlight" button to turn on a headlight if the headlight is off.



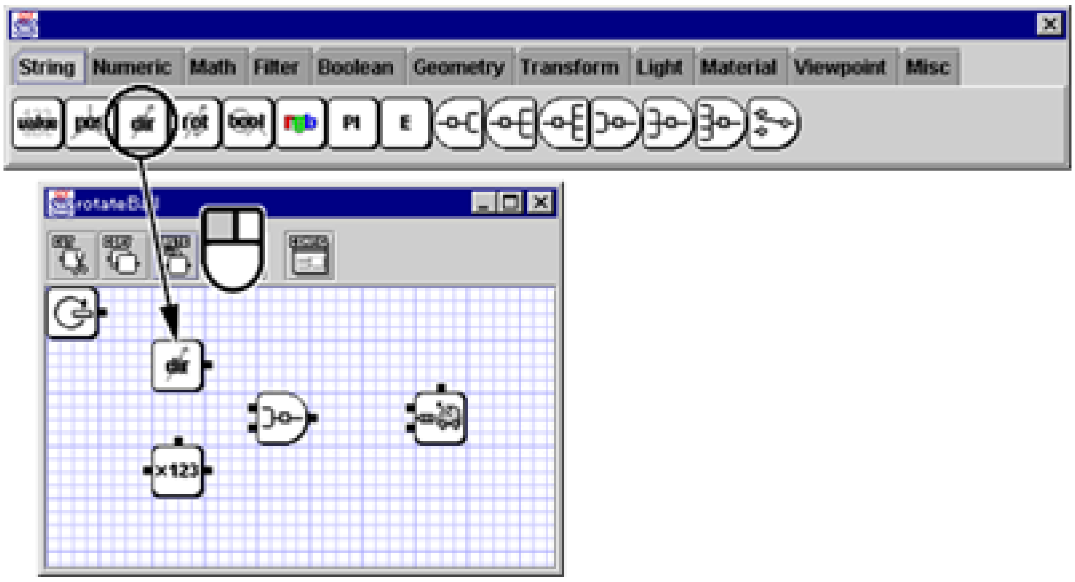
## 2. Creating a diagram to rotate the object

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

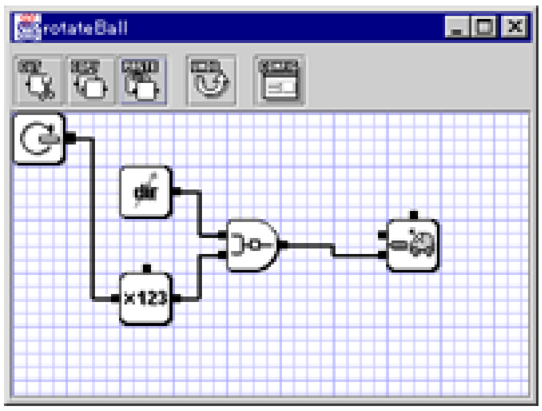


Next, drop the following four modules into the diagram from Module window,

|  | Icon | Class | Name |
|---|---|---|---|
| 1 | dir | String | Direction |
| 2 | ×123 | Filter | Scale |
| 3 |  | String | Mearge2Values |
| 4 |  | Transform | SetRotation |

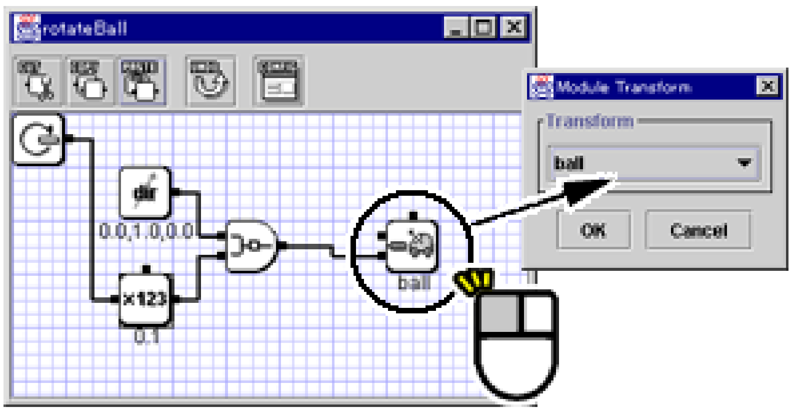


Then connect data-flow lines between the modules as following.



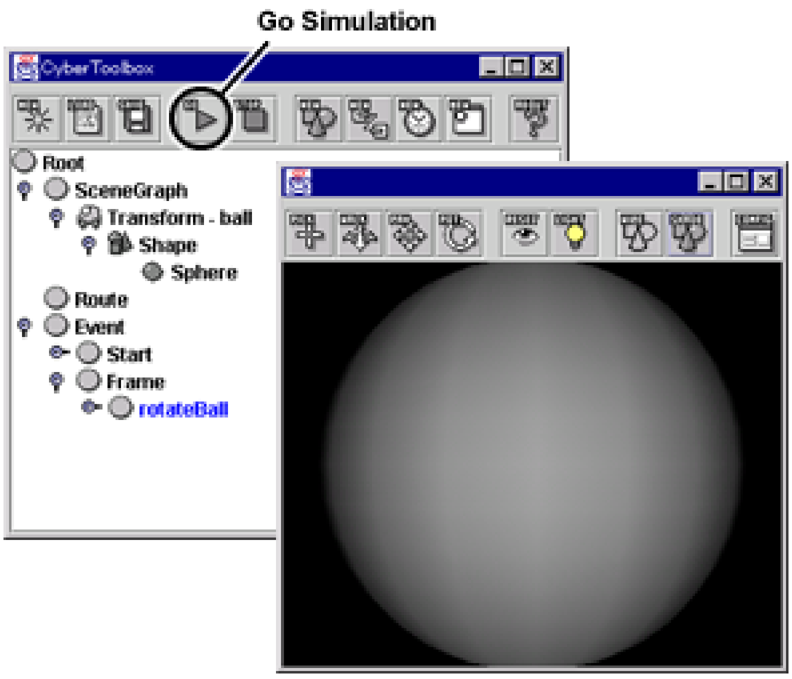
Finally, set internal values of the following module to double-click.

| | Icon | Specified value |
|---|---|---|
| 1 | dir | X=0, Y=1, Z=0 (0,1,0) |
| 2 | ×123 | 0.1 |
| 4 | | ball |



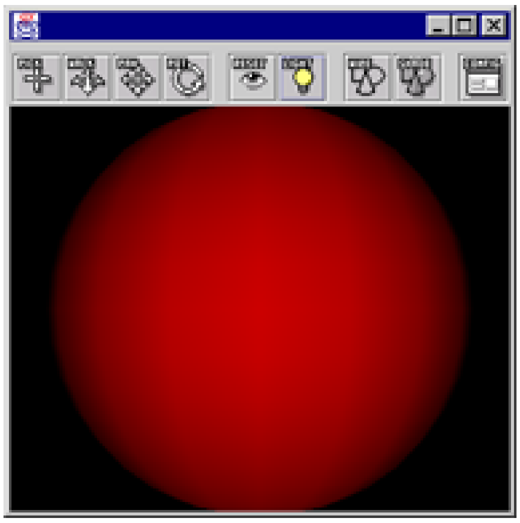
## 3. Starting the simulation

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



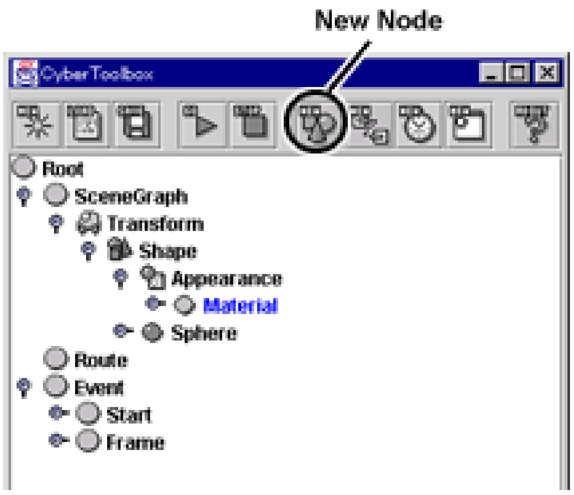
## *Flushing ball*

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

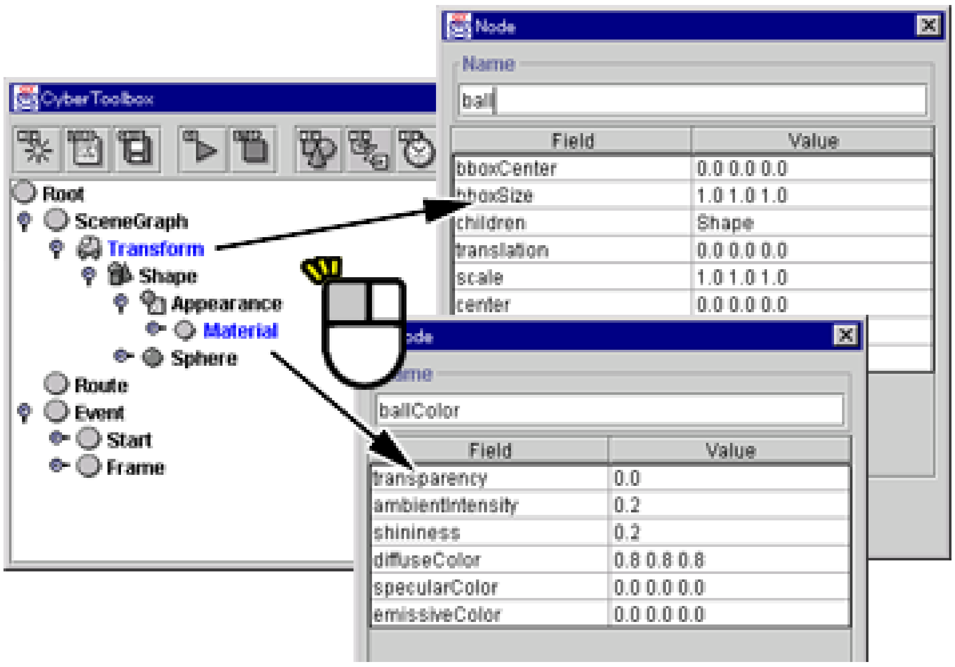


## *1. Creating a ball object with a material*

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. .



Click the added Shape node, and set "ballShape" to a name of the Shape node.. Similarly, set "ballColor" to the added Material node



ball

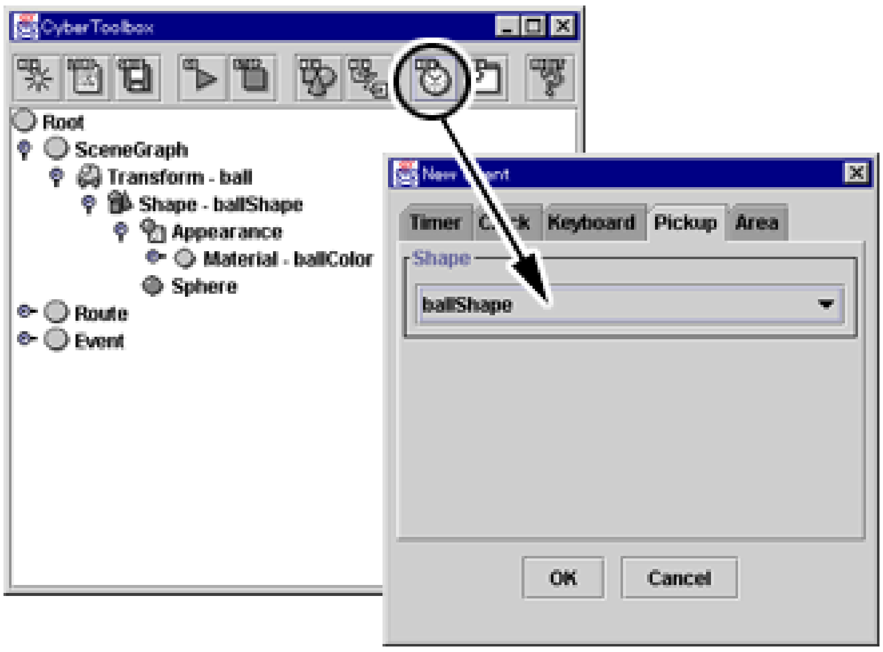
| Field | Value |
|---|---|
| bboxCenter | 0.0 0.0 0.0 |
| bboxSize | 1.0 1.0 1.0 |
| children | Shape |
| translation | 0.0 0.0 0.0 |
| scale | 1.0 1.0 1.0 |
| center | 0.0 0.0 0.0 |

ballColor

| Field | Value |
|---|---|
| transparency | 0.0 |
| ambientIntensity | 0.2 |
| shininess | 0.2 |
| diffuseColor | 0.8 0.8 0.8 |
| specularColor | 0.0 0.0 0.0 |
| emissiveColor | 0.0 0.0 0.0 |

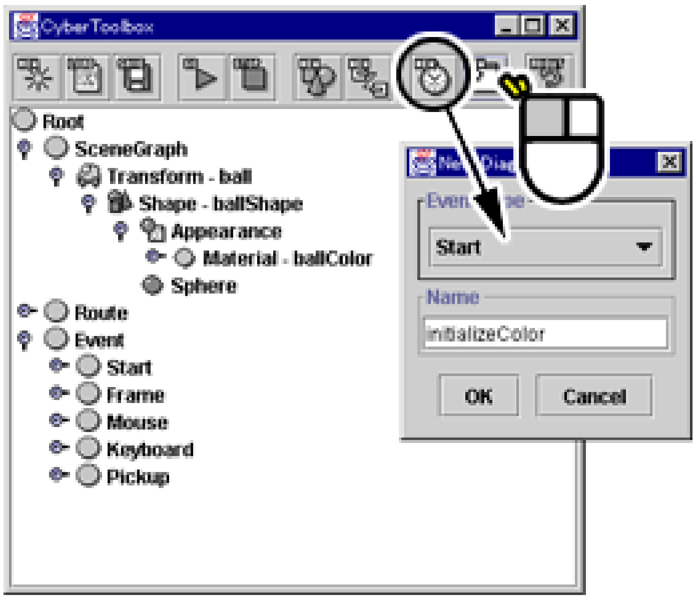
## *2. Creating a picking event*

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 "ballShape" in the list box, and click OK.



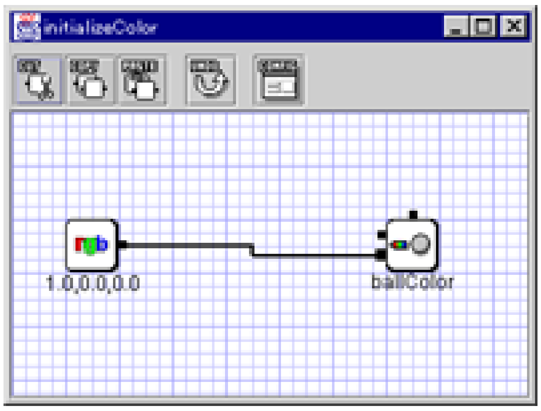
## 3. Creating a diagram to initialize the object color

To set a red color to the Material node when simulation is started, create a diagram that is run by Start event. To create the diagram, push “New Diagram” button in World window, select “Start” as the event type, and set “setColor” to the name.



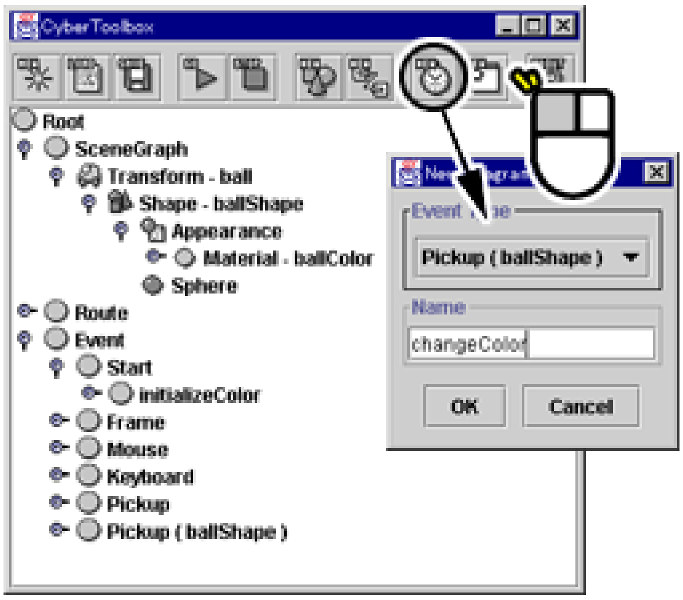
Then, add the following two modules into the diagram, set the module values, connect a data-flow line.

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



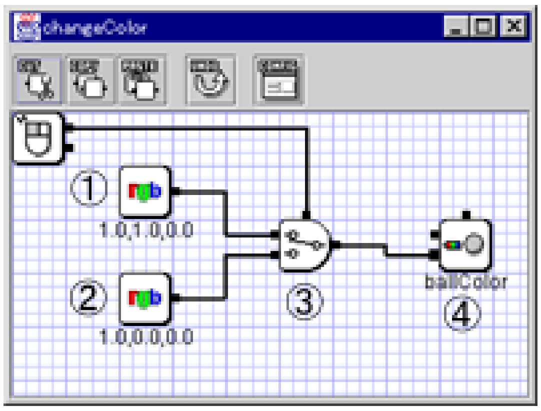
## *4. Creating a diagram to change the object color by clicking*

To set a yellow color to the Material node when the Shape node is clicked and set a red color when the Shape node is released, create a diagram that is executed by the picking event. To create the diagram, push "New Diagram" button in World window, select "Pickup (ballShape)" as the event type, and set "changeColor" to the name.



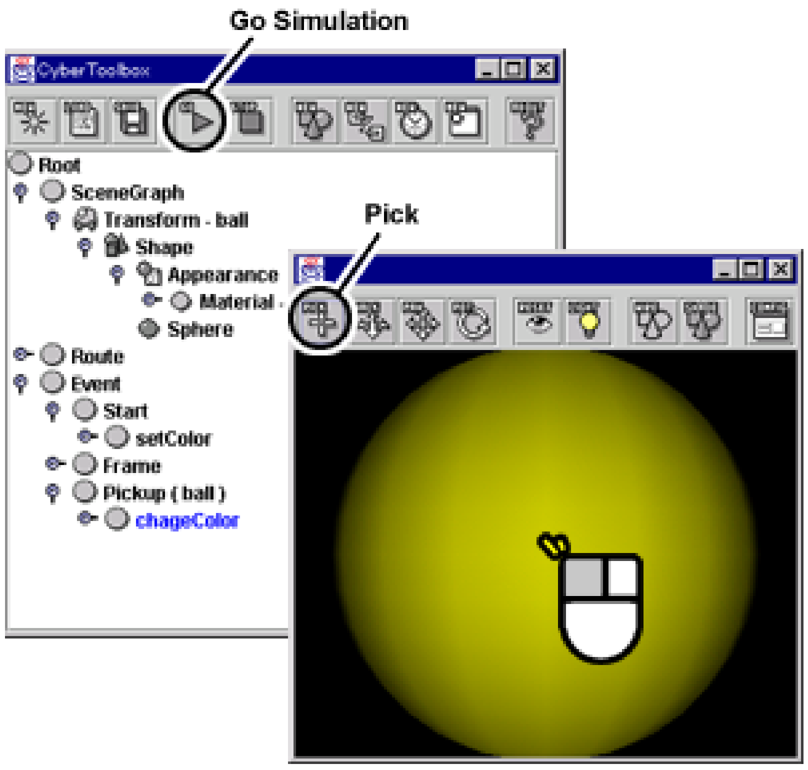
Then, add the following two modules into the diagram, set the module values, connect a data-flow line.

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



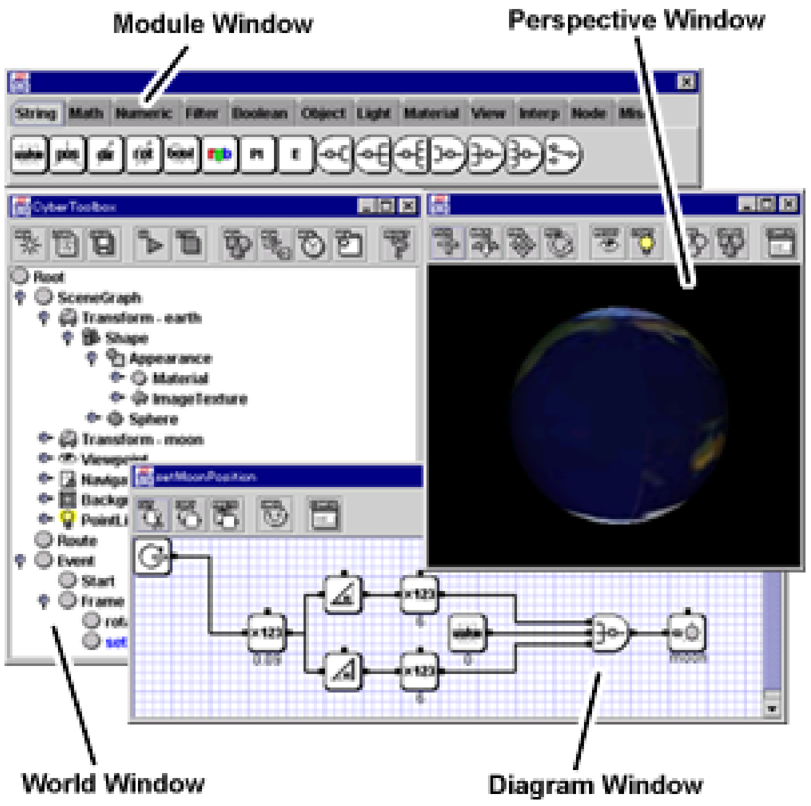
## 5. Starting the simulation

To check the behavior, click “Go Simulation” button in World window, and click “Pick” button in the Perspective window. When you click the ball in 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 world information that includes VRML node, route, event and diagram information. Using the window, you can edit all VRML and behavior information visually.

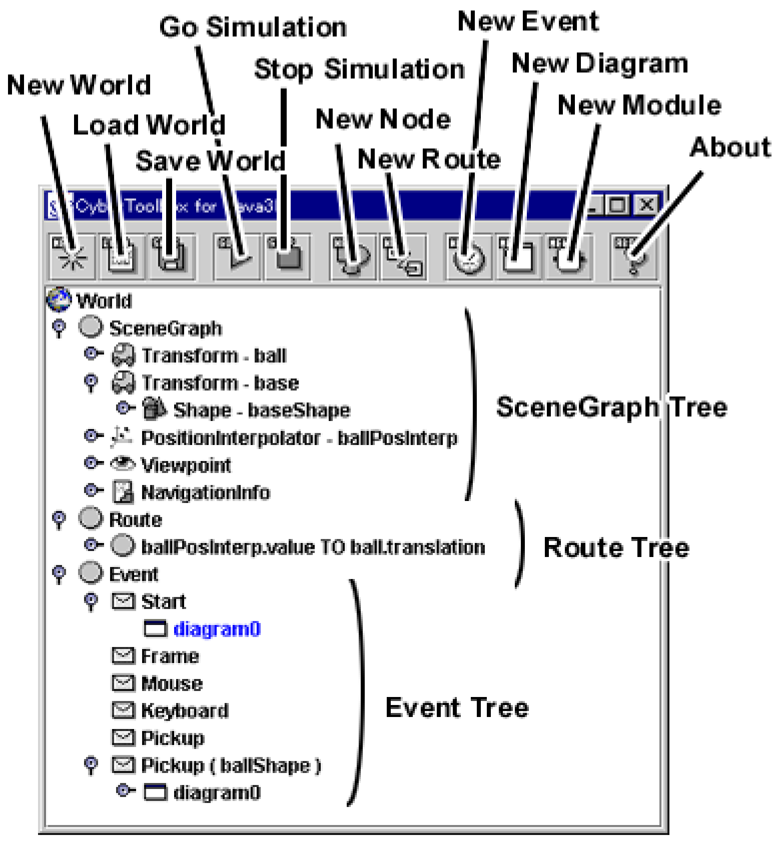
Perspective window shows graphical objects in the current world. You can check the world with the behaviors when the simulation is active, and walk in the world, and click the objects using a mouse.

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 and behavior information, allows you to read a geometry file to add the scene-graph information into a current world, save the current world information into a VRML97 file, create new nodes, events, diagrams which are workspaces to create behaviors, start and stop the simulation.

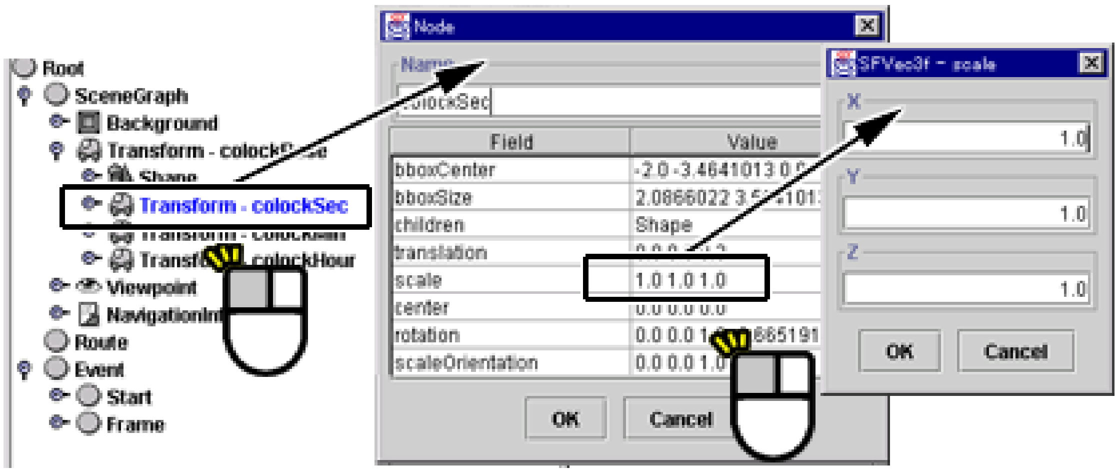
This window has three main trees, SceneGraph , Route , and Event tree.



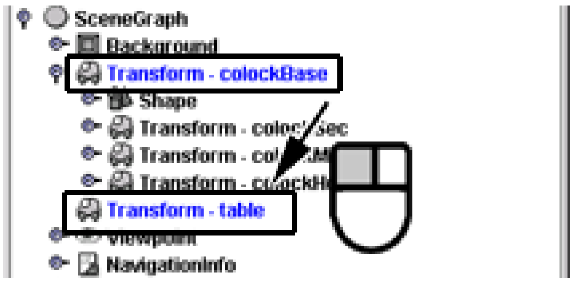
The SceneGraph tree shows all VRML node information, and the Route tree shows all VRML route information, and the Event tree shows all events and diagrams in the current world.

## SceneGraph Tree

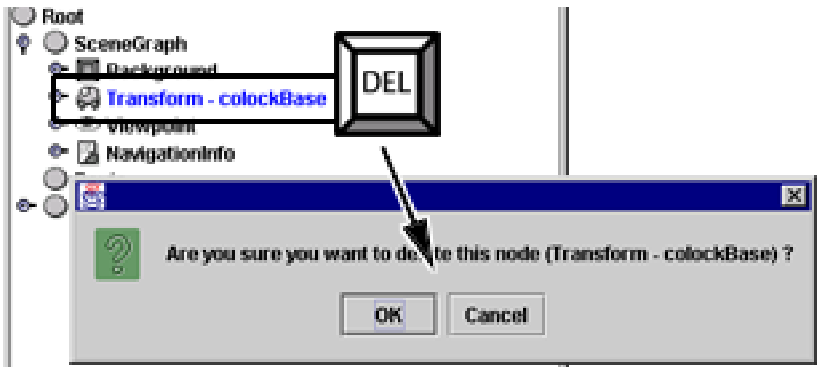
This tree shows all VRML node information in the current world. To confirm the node information in the tree, double-click the node item to open the setting dialog. To edit the field value, double-click the field in the dialog to open the dialog.



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.



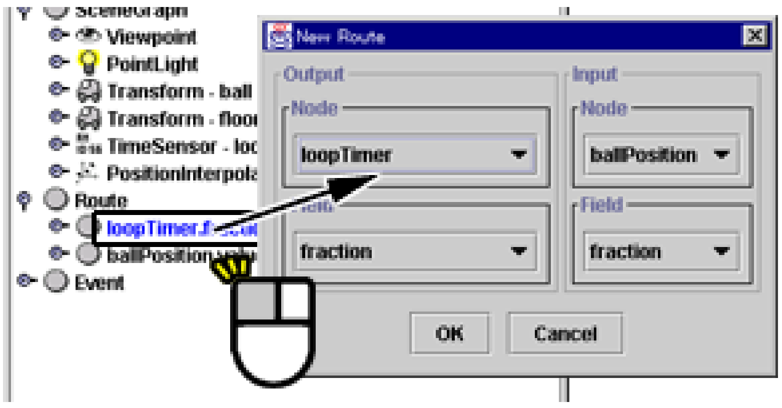
To delete a node, click the node item to select, then press the delete key. Click “Ok” button on the confirmation dialog if you sure want to delete it.



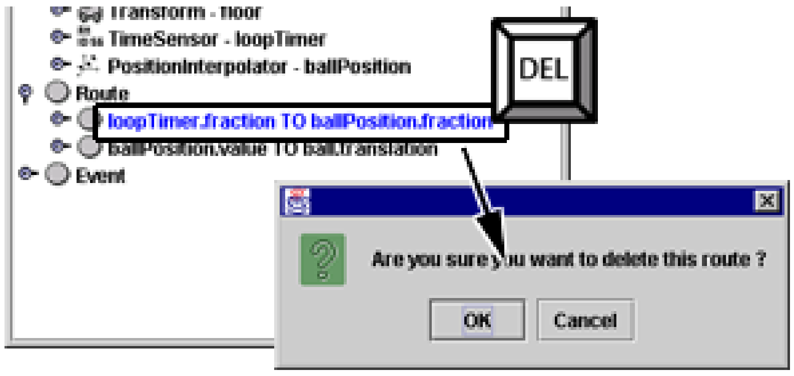
Use “New Node” button on the toolbar to add a new VRML node into the current world. See following “Toolbar” section about the button in more detail.

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

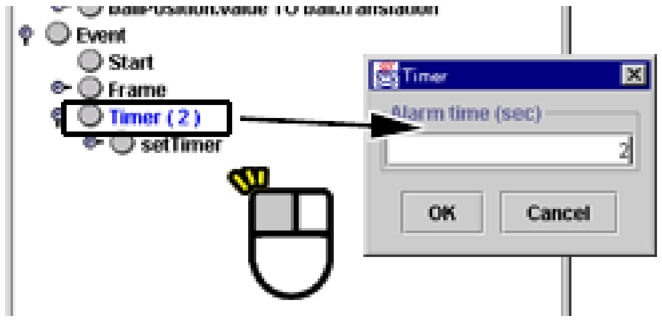

Transform - floor
TimeSensor - loopTimer
PositionInterpolator - ballPosition
Route
loopTimer.fraction TO ballPosition.fraction
ballPosition.value TO ball.translation
Event
DEL
Are you sure you want to delete this route ?
OK
Cancel

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.

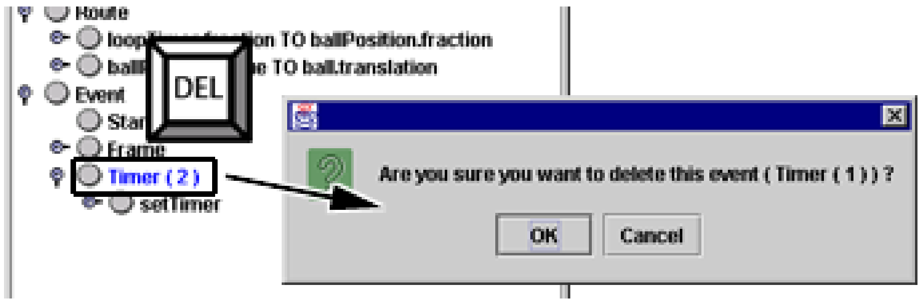
## Event Tree

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 user, 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.



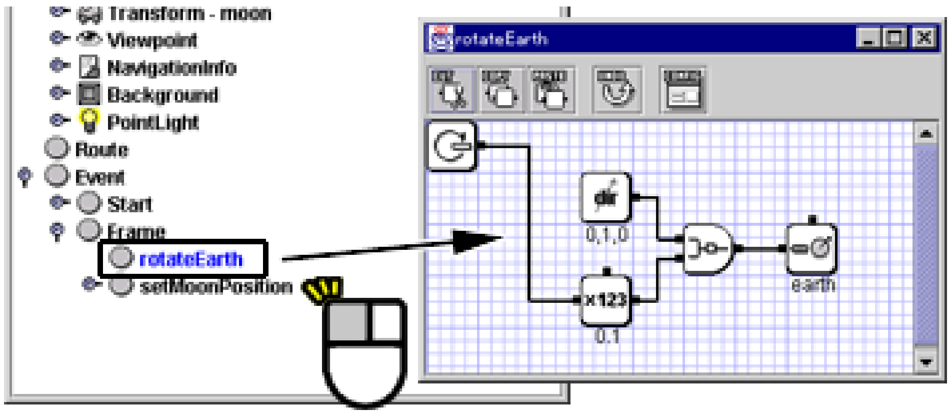
To delete a user event, click the event item, then press the delete key. Click “Ok” button on the confirmation dialog if you sure want to delete it. When the user event is deleted, the related diagrams are deleted too.



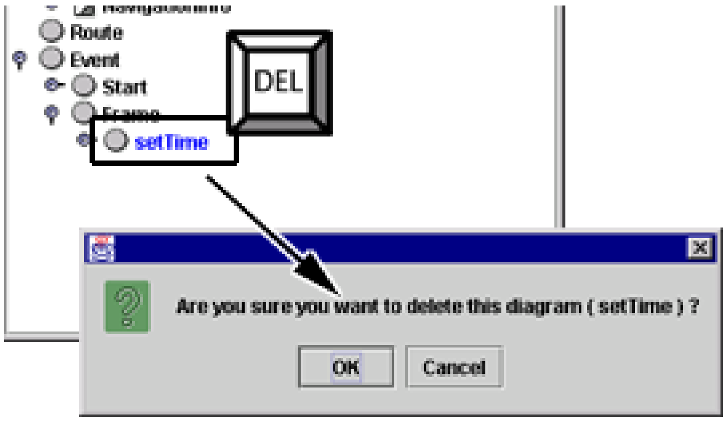
Are you sure you want to delete this event ( Timer ( 1 ) ) ?

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 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.



Are you sure you want to delete this diagram ( setTime ) ?

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, events and diagrams are deleted. The current world became empty.

### Load World

Click to load a geometry file, and add the all information into the current world. If the file format is VRML97 and it has behavior information, CyberToolbox load the behavior file too. See “Supported file formats” section about the supported file format

### Save World

Click to save the current world information into files. The scene graph information is saved intoa a VRML 97 (*.wrl), and the behavior information is saved into a original behavior file (*.cbf).

### Go Simulation

Click to start the current simulation to run 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.

### New Module

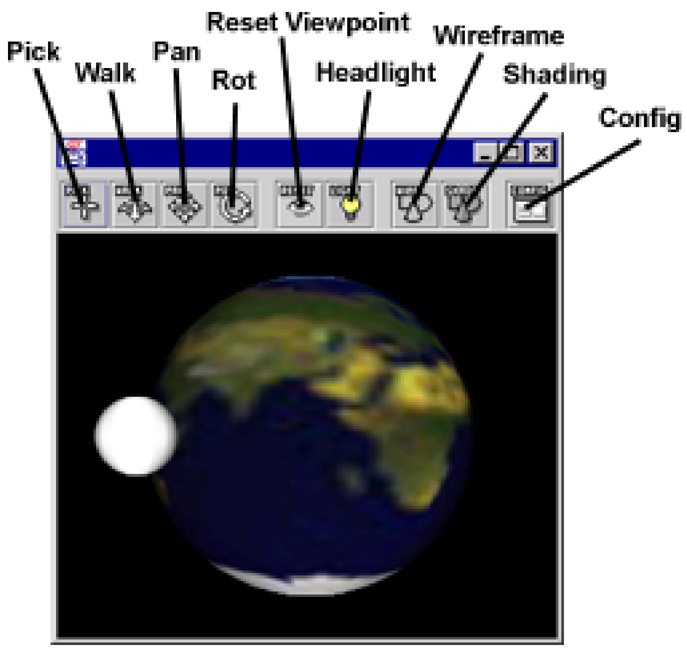
Click to add a new module into CyberToolbox. See “Adding your original modules” section about the user module in more detail.

### About

Click to see CyberToolbox information.

## Perspective Window

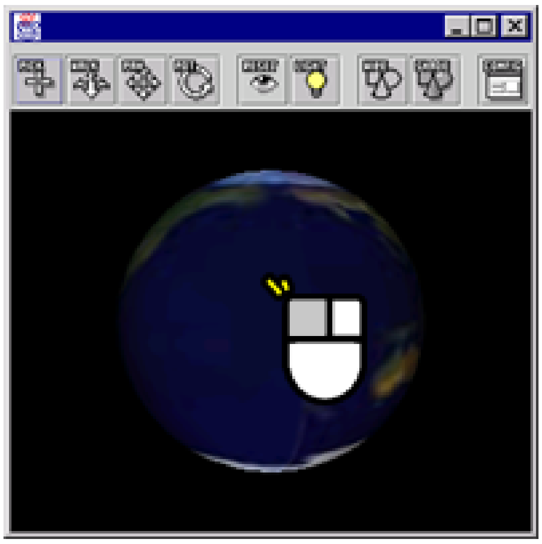
This window shows the current virtual world using Java3D. When the simulation is active, the window is updated with the behaviors. Using a mouse, you can move in the world and pick objects.



## Toolbar

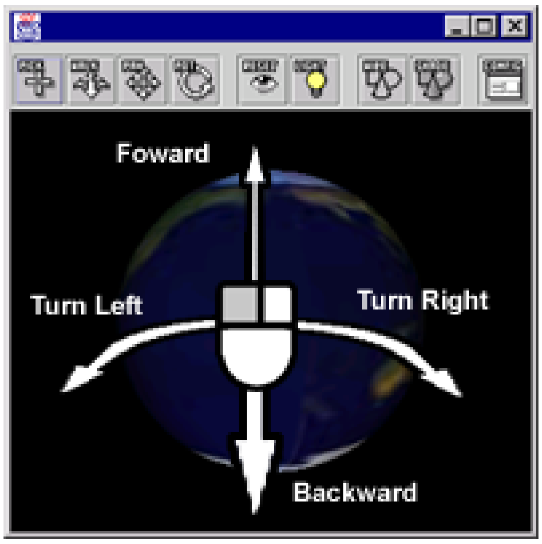
**Pick**

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



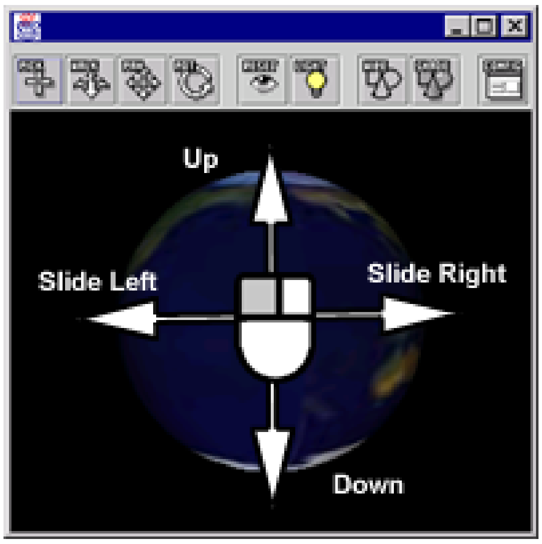
### Walk

Click to switch into Walk mode. Walk mode allows you to walk in the world using a mouse. You can move forward when the left button is pressed and the 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.



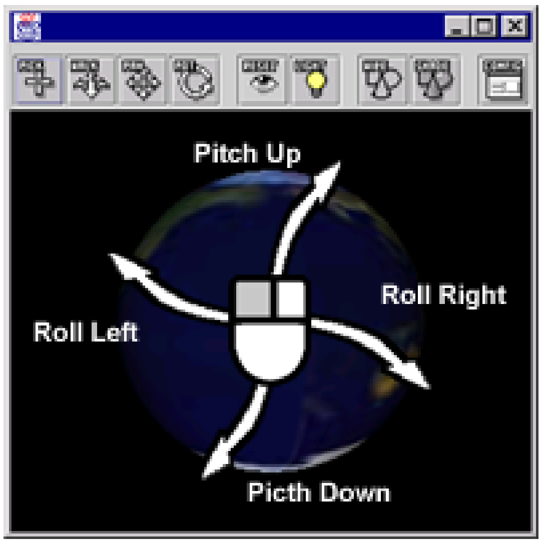
### Pan

Click to switch into Pan mode. Pan mode allows you to translate a current viewpoint vertically or horizontally using a mouse. You can up when the left button is pressed and the 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.



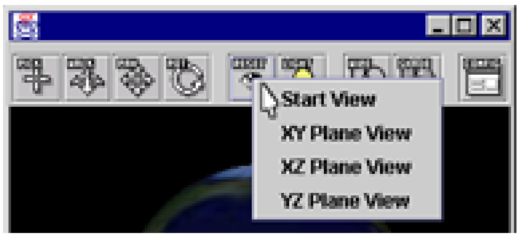
## Rot

Click to switch into Rot mode. Rot mode allows you rotate a current viewpoint using a mouse. You can pitch up when the left button is pressed and the 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

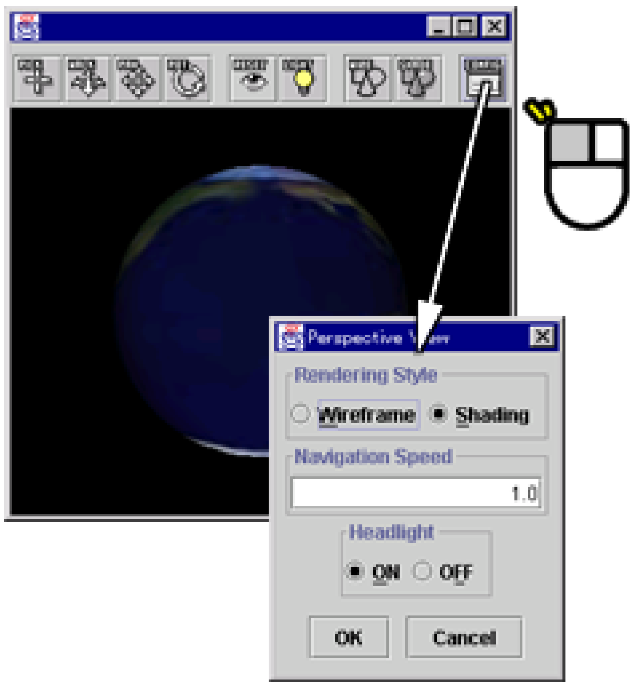
Click to change the rendering style into the wire-frame mode. When the rendering style is the wire-frame mode, all objects are rendered as wire-frame entities.

## 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.


Perspective View

Rendering Style: Wireframe [✓ Shading]

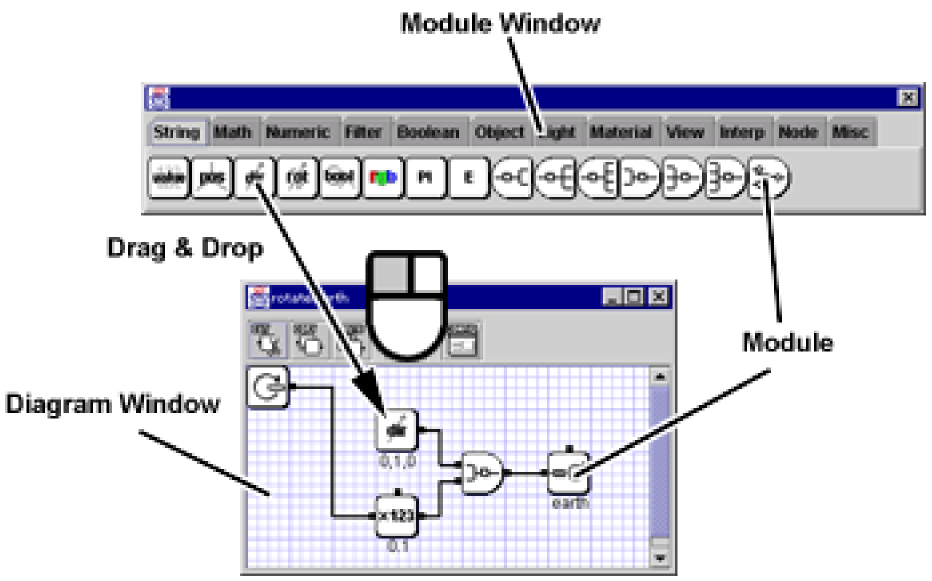
Navigation Speed: 1.0

Headlight: [✓ ON] OFF

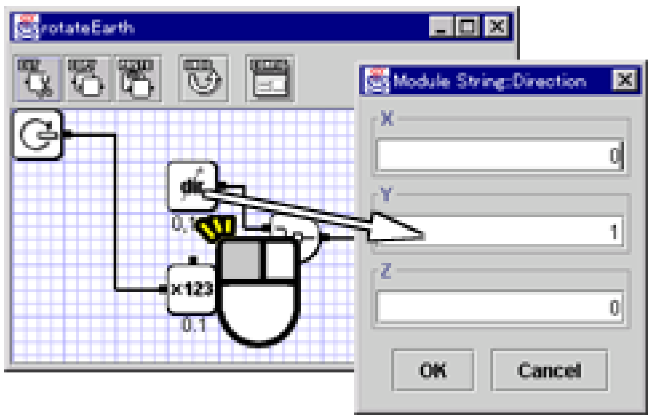
## Diagram / Module Window

The Diagram window is a workspace that you can create behaviors, and you can create the behaviors to connect between modules that are dropped form the Module window

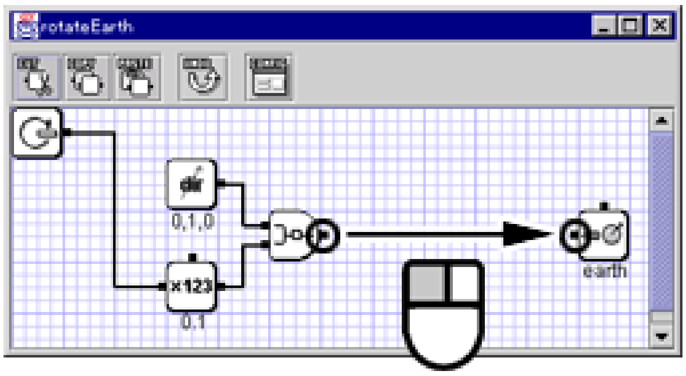
To add a new module into a diagram, drag the module in the Module window, and drop on the diagram window.



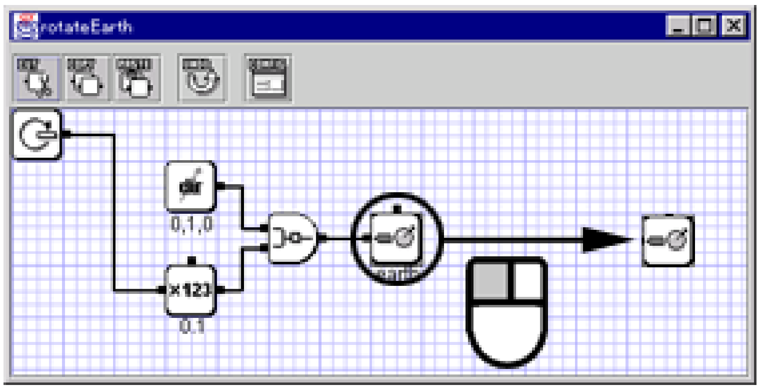
The module may have a setting dialog to set an inside value or a target node. To open the dialog, double-click the module.



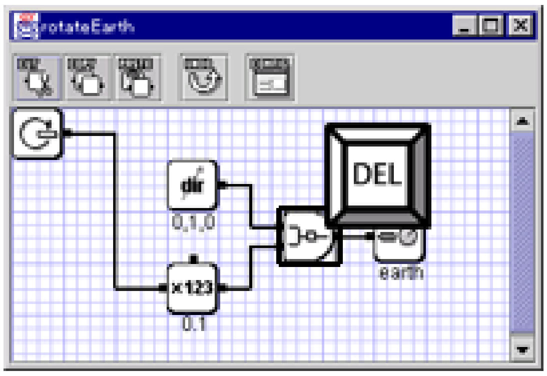
To add a data-flow line that connects between two module nodes, select the node, then drag the data-flow line, and drop on the other node.



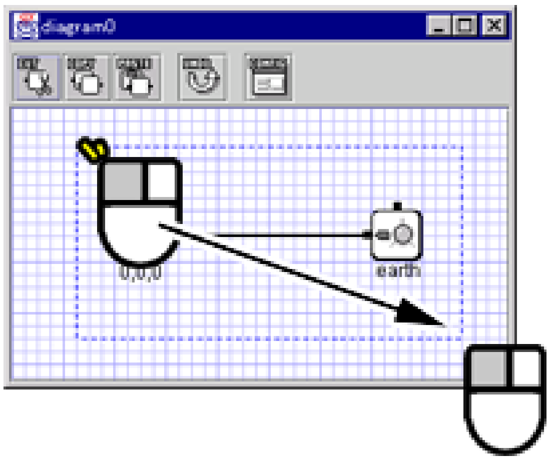
To move a module in a diagram window, drag the module.



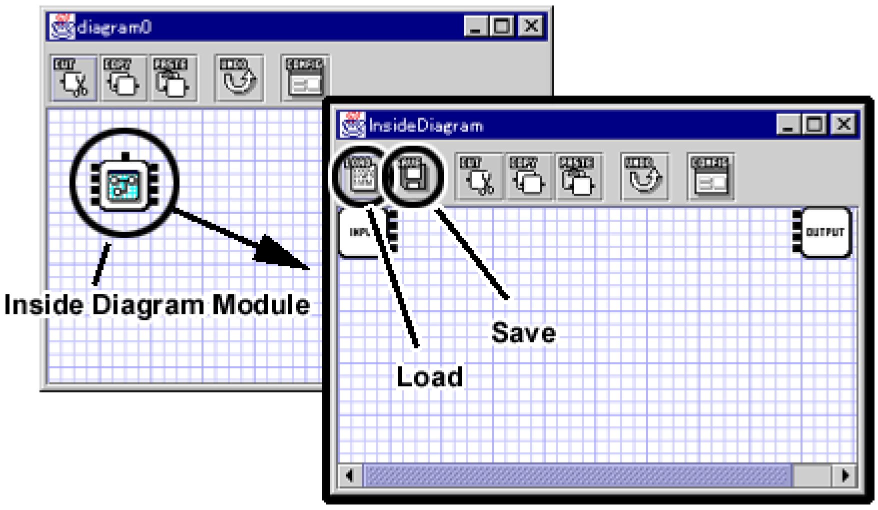
To delete a module or a data-flow line, click the module or the node, then push DEL key.



You can 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.



There is a module that can have a diagram inside the module. To edit the inside diagram, double-click the module to open the edit window. The edit window has “Load” and “Save” button specially to input or output the diagram define. Use the “Load” button to load a diagram define from a file. Use the “Save” button to save the current diagram define into a file. The default file extension is “*.dgm”.



## Toolbar

**Load**

Click to load a diagram define from a file. Only inside diagram has this button.

**Save**

Click to save a current diagram define into a file. Only inside diagram has this button.

**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.

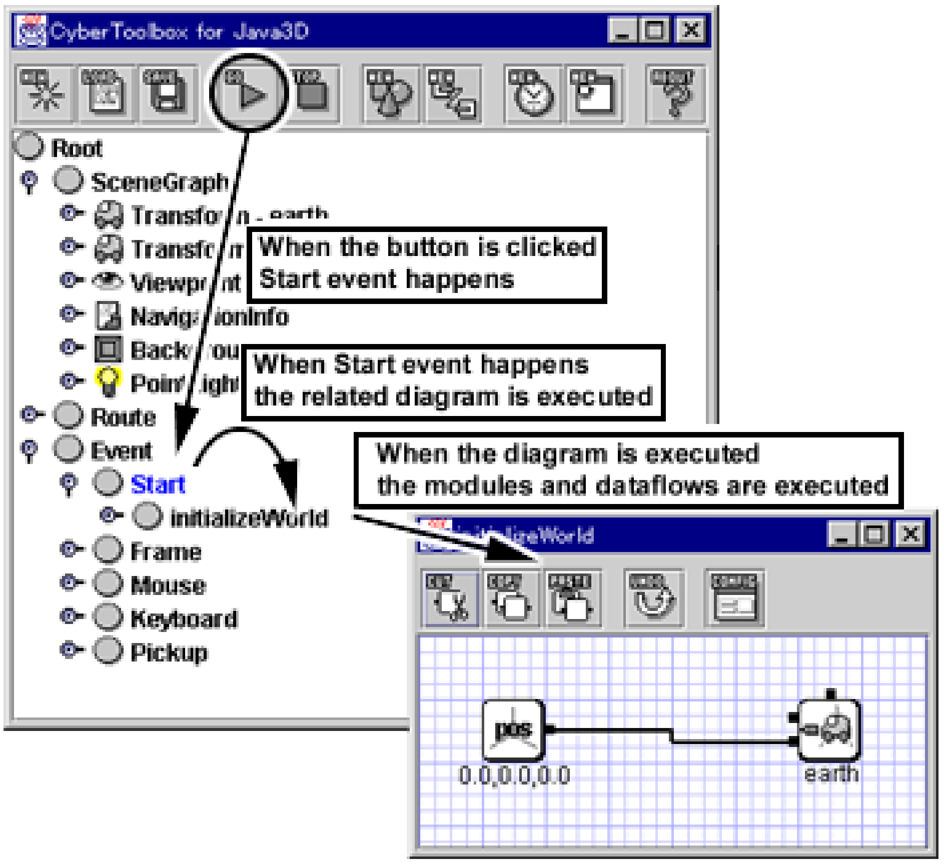
## Behavior Overview

CyberToolbox has a visual programming language to create behaviors in the world. Using the special language, everyone can create good behaviors in the world easily.

The behaviors are run by triggers that happens in the world.. The trigger is called as event in CyberToolbox. For example, Start event happens when the simulation is started, Pick event happens when a shape is clicked using a mouse.

When an event happens, diagrams that relate to the event are executed. The diagram has modules and data-flows to define the behaviors, and the modules and data-flows are executed when the diagrams are executed.

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

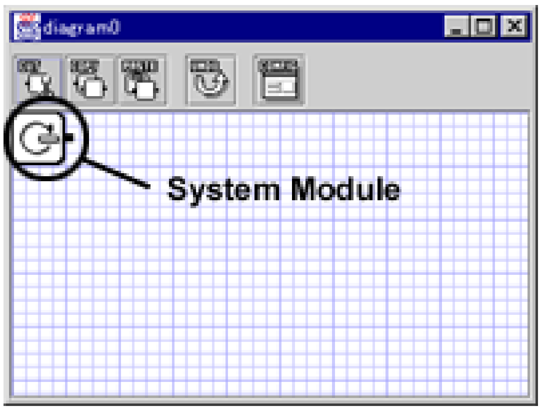


## Event

Currently, CyberToolbox has five system events and seven user events. The system events are added at first by CyberToolbox, you can add your original events using the user events.

| | Name | Occurrence Condition |
|---|---|---|
| System | Start | The simulation is started. |
| | Frame | Each frame |
| | Mouse | Mouse position or button is changed |
| | Keyboard | A key is pressed or released. |
| | Pickup | A shape is picked. |
| User | Timer | The specified time is exceeded. |
| | Clock | Each the specified time. |
| | Keyboard | The specified key is pressed or released |
| | Pickup | The specified shape is picked |
| | Collision | Specified two shapes are intersected. |
| | Area | User enters or exits in the specified region in the world. |

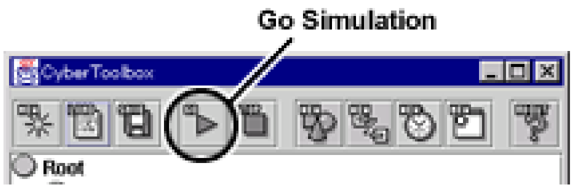
When you add a diagram, the diagram may have a system module as default. The module outputs information of the event that relates to the diagram into the output nodes. You can't remove the module from the diagram.



## ***System Event***

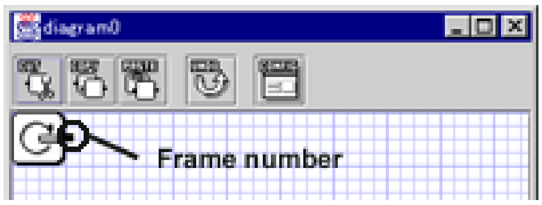
### Start

This event happens at once when the simulation is started to click "Go Simulation" button in World window. Use the event when you want to create behaviors for initialization purposes. The diagrams that relate to the event have no system module.



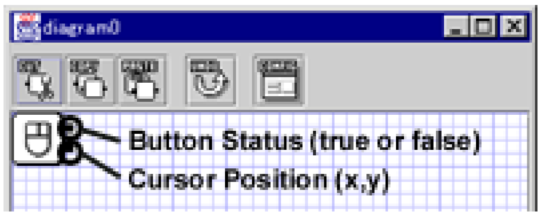
### Frame

This event happens at ten times per second after the simulation is started. The diagrams that relate to the event have a system module that outputs a current frame number.



### Mouse

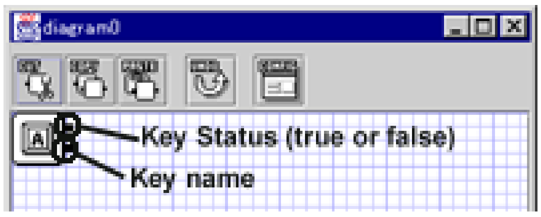
This event happens when a mouse is clicked and released on the perspective window. The diagrams that relate to the event have a system module that outputs the button status and the cursor position. When the mouse is dragged, the cursor position is updated.



To click the system module, you can choose an output coordinate format of the cursor position. If you want to get the normalized position, choose the "Normalized" option. The default output format is the frame coordinate.

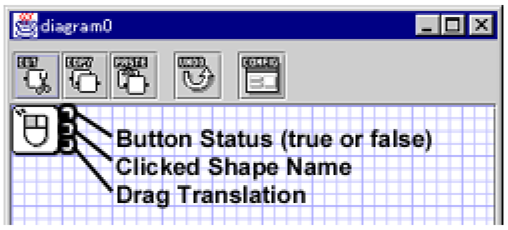
## Keyboard

This event happens when a key is pressed or released on the perspective window. The related diagram has a system module as default, and the module output the key status and the name of the clicked key.



## Pickup

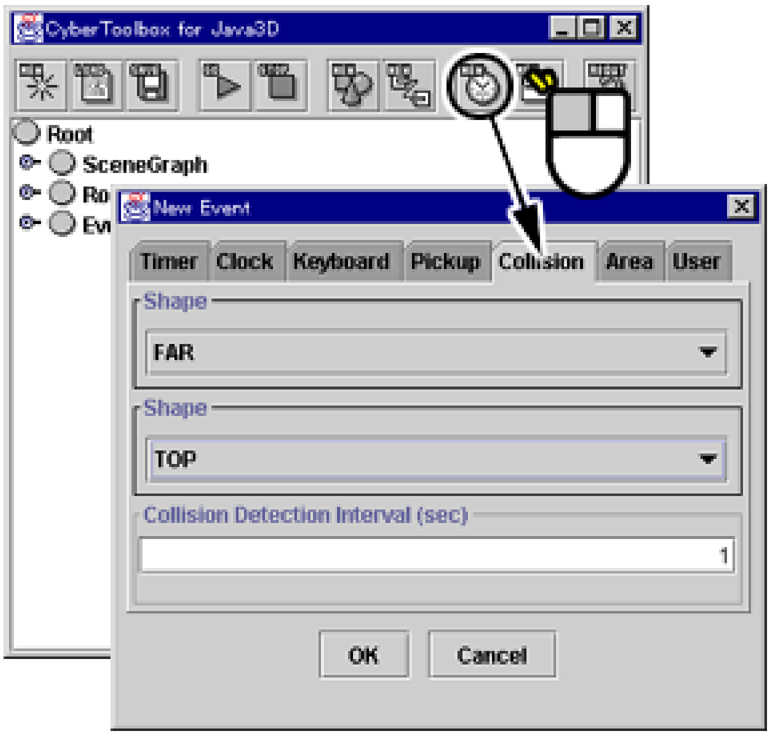
This event happens when a shape node in the scene graph is clicked or released on the perspective window. The diagrams that relate to the event have a system module that outputs a button status, a name of the clicked shape node and the drag translation.



To click the system module, you can choose an output coordinate format of the drag translation. If you want to get the normalized translation, choose the “Normalized” option. The default output format is the frame coordinate.

## User Event

CyberToolbox has seven user event types that you can specify the option parameters to create your original events. To add the new user event, click “New Event” button in World window, then select the event tab, set the option parameters, and click OK.



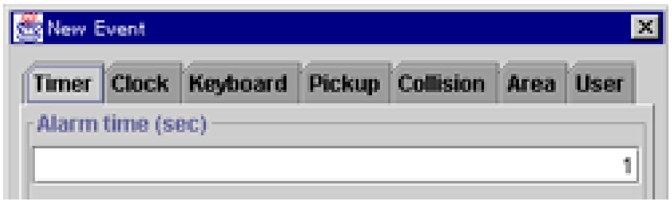
Shape: FAR

Shape: TOP

Collision Detection Interval (sec): 1

### Timer

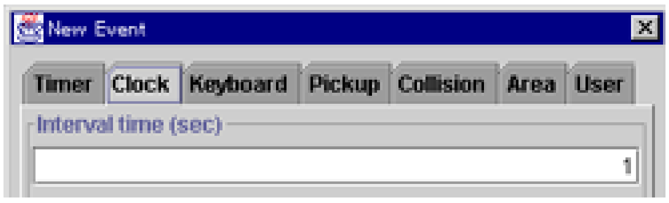
This event happens when the specified time is exceeded. The diagrams that relate to the event have no system module.



Alarm time (sec): 1

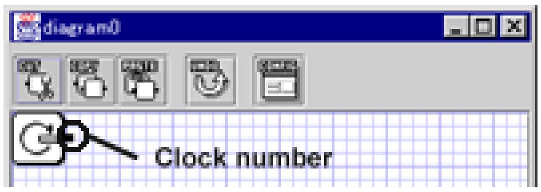
## Clock

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



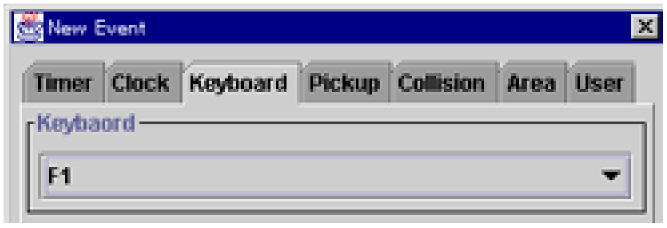
Interval time (sec): 1

The diagrams that relate to the event have a system module that outputs a clock number.



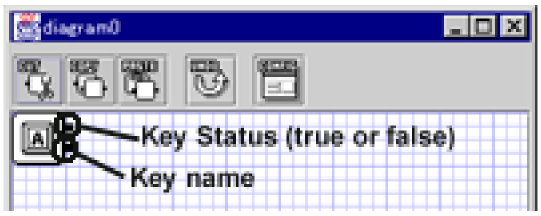
## Keyboard

This event happens only when the specified key is pressed or released on the perspective window.



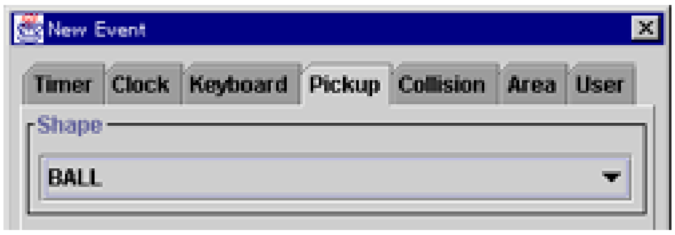
Keybaord: F1

The diagrams that relate to the event have a system module that outputs the key status and the name of the clicked key.



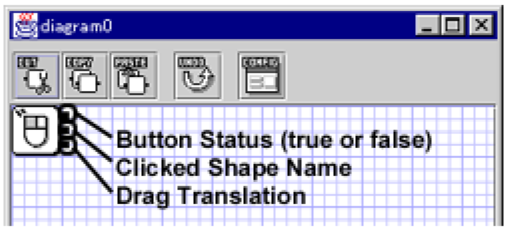
## Pickup

This event happens when the specified shape node in the scene graph is clicked or released on the perspective window. If the other shape node is clicked, the event doesn't happen.



Shape: BALL

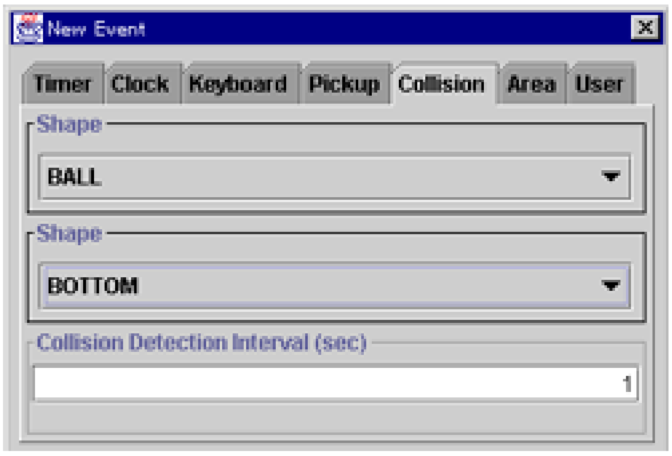
The diagrams that relate to the event have a system module that outputs a button status, a name of the clicked shape node and the drag translation.



To click the system module, you can choose an output coordinate format of the drag translation. If you want to get the normalized translation, choose the "Normalized" option. The default output format is the frame coordinate.

## Collision

This event happens when specified two shapes are intersected.

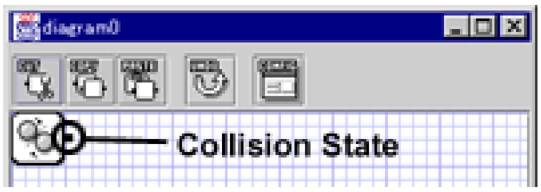


Shape: BALL

Shape: BOTTOM

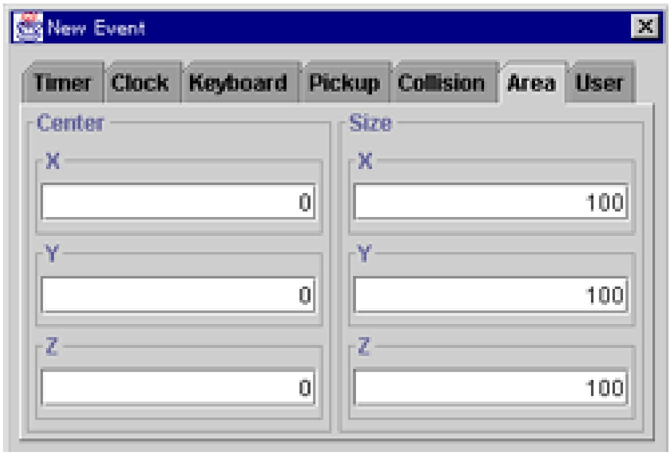
Collision Detection Interval (sec): 1

The diagrams that relate to the event have a system module that outputs a collision status. The module outputs a "true" string when the shapes are intersected, then the module outputs a “false” sting when the shapes are not intersected.



## Area

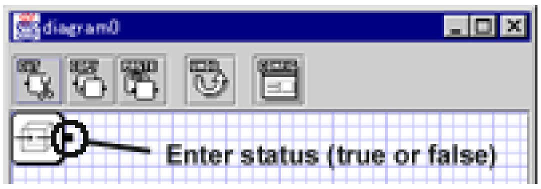
This event happens when you enter or exit in the specified region in the world. To add the new event, you have to set the center and the size.



Center
X 0
Y 0
Z 0

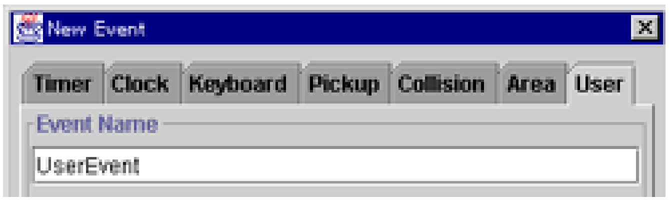
Size
X 100
Y 100
Z 100

The diagrams that relate to the event have a system module that outputs a “true” string when you enter the region, and outputs a “false” string when you exit the region.



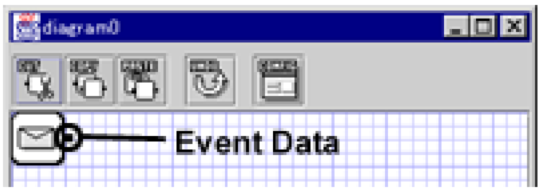
## User

This event happens when a message is sent to the event from a module of “Misc::SendMessage “.



Event Name
UserEvent

The diagrams that relate to the event have a system module that outputs a message that is send from the module.



## 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 sends a data from the output node to the input node of the other module along the data-flow line.

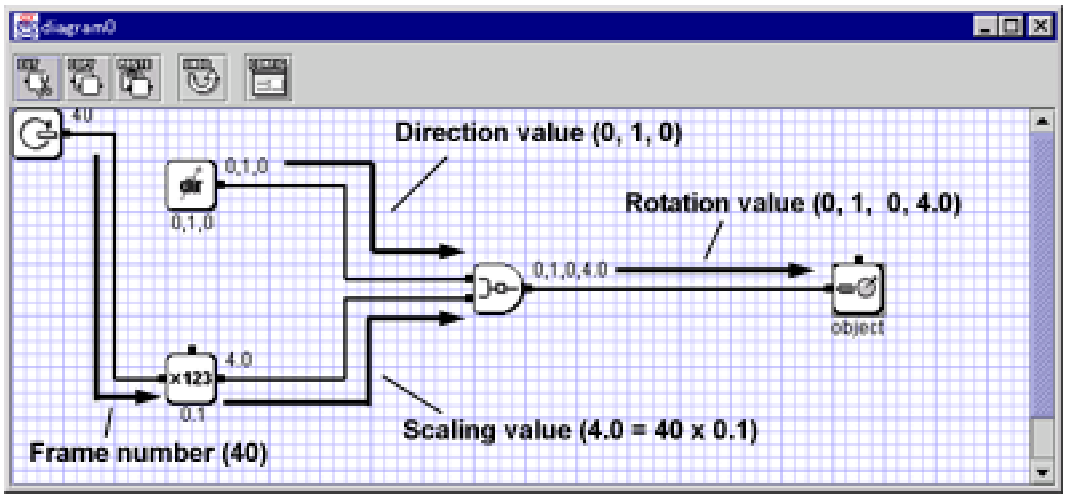
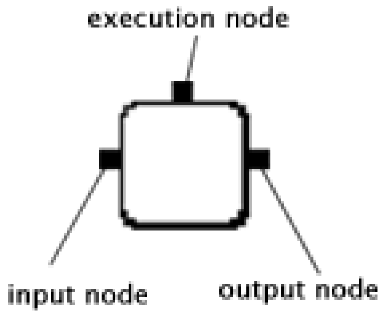


diagram0
40
0,1,0
0,1,0
Direction value (0, 1, 0)
Rotation value (0, 1, 0, 4.0)
0,1,0,4.0
object
4.0
x123
0.1
Frame number (40)
Scaling value (4.0 = 40 x 0.1)

The most top module in the data-flow is executed at first, the module sends data from the output nodes to other modules that are connected the data-flow lines with the output nodes. Similarly, the modules that receive the data from the top module sends data to other modules along the data-flow lines.

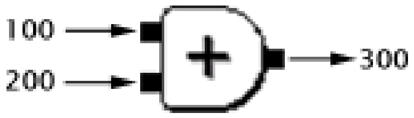
## Module

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



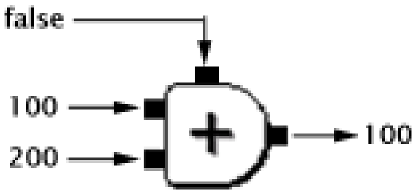
The input node receives a data from an output node from other module, the output node sends a data that is generated from the received data or the inside data.

For example, a following module has two input nodes and an output node., and the output result is a value that is added the two input data.



Using the execution node, you can activate or inactivate the module. If the execution node is not connected with a data-flow line from other module, the module is executed. When the execution node is connected and the input data is a “true” string, the module is executed. When the execution node is connected and the input data is not a “true” string, the module isn’t executed.

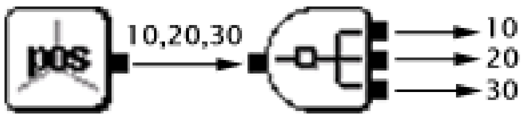
For example, a following module has two input nodes, an output node and an execution node and the execution node receives a “false” string. The output node sends the same value as the first input node because the module isn’t executed.



The data format of the all nodes is string. When a module has to calculate a string data of the nodes as a number, the module converts the string data into a number, then the module start the calculation.

The string data can include an information of some data to merge the data into a string using commas (‘,’). You can merge some data into a string, or divide a string into some data strings using modules of String class.

For example, a following left module outputs a position string which includes three numbers, and the right module divides the string into three number string.



## Module Behaviors

Modules are classified into some classes, String, Numeric, Math, Filter, Boolean, Geometry, Transform, Material, Light, Viewpoint, Interpolator, Sensor, Misc.

| Class Name | Function |
|---|---|
| String | Outputs the specified value. Merges or splits the input string, etc. |
| Numeric | Adds or subtracts two input values, etc … |
| Math | Increments the input value, gets the absolute value, etc. |
| Filter | Scales the input value, Check a range of the input value, etc |
| Boolean | Check whether a first input value is greater than a second input value, etc |
| Geometry | Gets a length of the input vector, rotates the input vector, etc. |
| Transform | Sets or gets a field value of a Transform node |
| Material | Sets or gets a field value of a Material node |
| Light | Sets or gets a field value of a DirectionalLight, PointLight or SpotLight node |
| Viewpoint | Sets or gets a field value of a Viewpoint node |
| Interpolator | Play an Interpolator node as an animation sequence, etc. |
| Sensor | Get a record from VR sensors, Polhemus Fastrak etc. |
| Misc | Sets or gets a global data, etc… |

## String

### Value

This module outputs the specified string into the output node.

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

### Position

This module outputs the specified position infomation into the output node.

| | |
|---|---|
| Input node names | - |

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

## Direction

This module outputs the specified direction infomation into the output node.

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

## Rotation

This module outputs the specified rotation information into the output node.

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

## Bool

This module outputs the specified boolean information into the output node.

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

## Color

This module outputs the specified color information into the output node.

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

## PI

This module outputs a ratio of the circumference of a circle value into the output node.

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

## E

This module outputs a base of the natural logarithms into the output node.

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

## Gravity

This module outputs a gravity value (=9.8).

| | |
|---|---|
| Input node names | - |
| Output node names | OutValue |
| Execution node | - |

| | |
|---|---|
| Setting dialog | - |
| Target node | - |
| Result | OutValue =  9.8 |

## Divide2Values

This module divides the input string into two strings, and outputs the divided strings to output nodes.

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

## Divide3Values

This module divides the input string into three strings, and outputs the divided strings to output nodes.

| | |
|---|---|
| Input node names | InValue (value1,value2, value3) |
| Output node names | OutValue1<br>OutValue2<br>OutValue3 |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | OutValue1 = value1<br>OutValue2 = value2<br>OutValue3 = value3 |

| | |
|---|---|
| Example | InValue = 100,200,300<br>OutValue1 = 100<br>OutValue2 = 200<br>OutValue3 = 300 |

## Divide4Values

This module divides the input string into four strings, and outputs the divided strings to output nodes.

| | |
|---|---|
| Input node names | InValue (value1,value2, value3,value4) |
| Output node names | OutValue1<br>OutValue2<br>OutValue3<br>OutValue4 |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | OutValue1 = value1<br>OutValue2 = value2<br>OutValue3 = value3<br>OutValue4 = value4 |
| Example | InValue = 100,200,300,400<br>OutValue1 = 100<br>OutValue2 = 200<br>OutValue3 = 300<br>OutValue4 = 400 |

## Merge2Values

This module merges two input strings into a string, and outputs the merged string to an output node.

| | |
|---|---|
| Input node names | InValue1<br>InValue2 |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |

| | |
|---|---|
| Result | OutValue = InValue1,InValue2 |
| Example | InValue1 = 100<br>InValue2 = 200<br>OutValue1 = 100,200 |

### Merge3Values

This module merges three input strings into a string, and outputs the merged string to an output node.

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

### Merge4Values

This module merges four input strings into a string, and outputs the merged string to an output node.

| | |
|---|---|
| Input node names | InValue1<br>InValue2<br>InValue3<br>InValue4 |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | OutValue = InValue1,InValue2,InValue3,InValu4 |

| | |
|---|---|
| Example | InValue1 = 100<br>InValue2 = 200<br>InValue3 = 300<br>InValue4 = 400<br>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 a string of the first input node into the output node. Otherwise, outputs a string of the second input node.

| | |
|---|---|
| Input node names | InValue1<br>InValue2 |
| Output node names | OutValue |
| Execution node | O |
| Setting dialog | - |
| Target node | - |
| Result | if (ExecutionNode is not connected)<br>    OutValue = InValue1<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = InValue1<br>    else<br>        OutValue = InValue2<br>} |
| Example | InValue1 = 100<br>InValue2 = 200<br>ExecutionNode = “false”<br>OutValue = 200 |

## Numeric

All modules of this class have an execution node. When the execution node is not connected with a data-flow line or the execution node receives a “true” string, the modules output a calculation result into output nodes. Otherwise, the modules output values of the input nodes without the calculation.

### Add

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

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

## Sub

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

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

## Mul

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

| | |
|---|---|
| Input node names | InValue1<br>InValue2 |
| Output node names | OutValue |
| Execution node | O |
| Setting dialog | - |
| Target node | - |

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

### ÷ **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<br>InValue2 |
| Output node names | OutValue |
| Execution node | O |
| Setting dialog | - |
| Target node | - |

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

### Mod

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

| | |
|---|---|
| Input node names | InValue1<br>InValue2 |
| Output node names | OutValue1<br>OutValue2 |
| Execution node | O |
| Setting dialog | - |
| Target node | - |

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

## And

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

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

| | |
|---|---|
| Example | InValue1 = 1<br>InValue2 = 2<br>OutValue = 0 |

## Or

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

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

## Xor

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

| | |
|---|---|
| Input node names | InValue1<br>InValue2 |
| Output node names | OutValue |
| Execution node | O |
| Setting dialog | - |
| Target node | - |

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

## Math

All modules of this class have an execution node. When the execution node is not connected with a data-flow line or the execution node receives a “true” string, the modules output a calculation result into output nodes. Otherwise, the modules output values of the input nodes without the calculation.

### Increment

This module adds 1 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 | - |
| Target node | - |
| Result | if (ExecutionNode is not connected)<br>    OutValue = InValue + 1<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = InValue + 1<br>    else<br>        OutValue = InValue<br>} |
| Example | InValue = 1.1<br>OutValue = 2.1 |

### Decrement

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

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

| | |
|---|---|
| Result | if (ExecutionNode is not connected)<br>    OutValue = InValue - 1<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = InValue - 1<br>    else<br>        OutValue = InValue<br>} |
| Example | InValue = 1<br>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 | if (ExecutionNode is not connected)<br>    OutValue = \| InValue \|<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = \| InValue \|<br>    else<br>        OutValue = InValue<br>} |
| Example | InValue = -1<br>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 | if (ExecutionNode is not connected)<br>    OutValue = - InValue<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = - InValue<br>    else<br>        OutValue = InValue<br>} |
| Example | InValue = 1<br>OutValue = -1 |

## 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 | InValue1<br>InValue2 |
| Output node names | OutValue |
| Execution node | O |
| Setting dialog | - |
| Target node | - |
| Result | if (ExecutionNode is not connected)<br>    OutValue = pow( InValue1, InValue2)<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = pow( InValue1, InValue2)<br>    else<br>        OutValue = InValue<br>} |

| | |
|---|---|
| Example | InValue1 = 2<br>InValue2 = 3<br>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 | if (ExecutionNode is not connected)<br>    OutValue = sqrt(InValue)<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = sqrt(InValue)<br>    else<br>        OutValue = InValue<br>} |
| Example | InValue = 9<br>OutValue = 3 |

## Min

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

| | |
|---|---|
| Input node names | InValue1<br>InValue2 |
| Output node names | OutValue |
| Execution node | O |
| Setting dialog | - |
| Target node | - |

| | |
|---|---|
| Result | if (ExecutionNode is not connected) {<br>    if (InValue1 < InValue2)<br>        OutValue = InValue1<br>    else<br>        OutValue = InValue2<br>    OutValue = sqrt(InValue)<br>}<br>else {<br>    if (ExecutionNode data is “true”) {<br>        if (InValue1 < InValue2)<br>            OutValue = InValue1<br>        else<br>            OutValue = InValue2<br>    }<br>    else<br>        OutValue = InValue<br>} |
| Example | InValue1 = 100<br>InValue2 = 200<br>OutValue = 100 |

## Max

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

| | |
|---|---|
| Input node names | InValue1<br>InValue2 |
| Output node names | OutValue |
| Execution node | O |
| Setting dialog | - |
| Target node | - |

| | |
|---|---|
| Result | if (ExecutionNode is not connected) {<br>    if (InValue1 > InValue2)<br>        OutValue = InValue1<br>    else<br>        OutValue = InValue2<br>    OutValue = sqrt(InValue)<br>}<br>else {<br>    if (ExecutionNode data is “true”) {<br>        if (InValue1 > InValue2)<br>            OutValue = InValue1<br>        else<br>            OutValue = InValue2<br>    }<br>    else<br>        OutValue = InValue<br>} |
| Example | InValue1 = 100<br>InValue2 = 200<br>OutValue = 100 |

## 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 | if (ExecutionNode is not connected)<br>    OutValue = log( InValue)<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = log( InValue)<br>    else<br>        OutValue = InValue<br>} |

## Exp

This module outputs an 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 | if (ExecutionNode is not connected)<br>    OutValue = exp( InValue)<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = exp( InValue)<br>    else<br>        OutValue = InValue<br>} |

## 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 | if (ExecutionNode is not connected)<br>    OutValue = sin( RadianAngle)<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = sin( RadianAngle)<br>    else<br>        OutValue = RadianAngle<br>} |

## 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 | if (ExecutionNode is not connected)<br>   OutValue = cos( RadianAngle)<br>else {<br>   if (ExecutionNode data is “true”)<br>      OutValue = cos( RadianAngle)<br>   else<br>      OutValue = RadianAngle<br>} |

## 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 | if (ExecutionNode is not connected)<br>   OutValue = tan( RadianAngle)<br>else {<br>   if (ExecutionNode data is “true”)<br>      OutValue = tan( RadianAngle)<br>   else<br>      OutValue = RadianAngle<br>} |

## 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 | if (ExecutionNode is not connected)<br>    OutValue = asin( InValue)<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = asin( InValue)<br>    else<br>        OutValue = InValue<br>} |

## 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 | if (ExecutionNode is not connected)<br>    OutValue = acos( InValue)<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = acos( InValue)<br>    else<br>        OutValue = InValue<br>} |

## 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 | if (ExecutionNode is not connected)<br>    OutValue = atan( InValue)<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = atan( InValue)<br>    else<br>        OutValue = InValue<br>} |

## 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 x 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

All modules of this class have an execution node. When the execution node is not connected with a data-flow line or the execution node receives a “true” string, the modules output a calculation result into output nodes. Otherwise, the modules output values of the input nodes without the calculation.

### Offset

This module adds the specified 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 | if (ExecutionNode is not connected)<br>    OutValue = InValue + User specified value<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = InValue + User specified value<br>    Else<br>        OutValue = InValue<br>} |
| Example | InValue = 10<br>User specified value = 1000<br>OutValue = 1010 |

### Mul

This module multiplies the input value by the specified 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 | if (ExecutionNode is not connected)<br>    OutValue = InValue * User specified value<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = InValue * User specified value<br>    Else<br>        OutValue = InValue<br>} |
| Example | InValue = 10<br>User specified value = 20<br>OutValue = 200 |

## Div

This module divides the input value by the specified 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 | if (ExecutionNode is not connected)<br>    OutValue = InValue * User specified value<br>else {<br>    if (ExecutionNode data is “true”)<br>        OutValue = InValue * User specified value<br>    Else<br>        OutValue = InValue<br>} |
| Example | InValue = 10<br>User specified value = 20<br>OutValue = 200 |

## 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 | if (ExecutionNode is not connected) |
| |     OutValue = ceil(InValue) |
| | else { |
| |     if (ExecutionNode data is “true”) |
| |         OutValue = ceil(InValue) |
| |     Else |
| |         OutValue = InValue |
| | } |
| Example | InValue = 12.3 |
| | OutValue = 13 |

## Floor

This module outputs the largest value that is not greater 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 | if (ExecutionNode is not connected) |
| |     OutValue = floor(InValue) |
| | else { |
| |     if (ExecutionNode data is “true”) |
| |         OutValue = floor(InValue) |
| |     else |
| |         OutValue = InValue |
| | } |

| | |
|---|---|
| Example | InValue = 12.3<br>OutValue = 12 |

## High

This module outputs the specified high value when the input value is greater than the specified 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 | if (ExecutionNode is not connected) {<br>    if (User specified high value < InValue)<br>        OutValue = User specified high value<br>    Else<br>        OutValue = InValue<br>}<br>else {<br>    if (ExecutionNode data is “true”) {<br>        if (User specified hi value < InValue)<br>            OutValue = User specified high value<br>        Else<br>            OutValue = InValue<br>    }<br>    else<br>        OutValue = InValue<br>} |
| Example | InValue = 120<br>User specified high value = 100<br>OutValue = 100 |

### Low

This module outputs the specified low value when the input value is less than the specified 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 | if (ExecutionNode is not connected) {<br>    if (InValue < User specified low value)<br>        OutValue = User specified low value<br>    else<br>        OutValue = InValue<br>}<br>else {<br>    if (ExecutionNode data is “true”) {<br>        if (InValue M ¥¥< User specified low data)<br>            OutValue = User specified log value<br>        else<br>            OutValue = InValue<br>    }<br>    else<br>        OutValue = InValue<br>} |
| Example | InValue = 12.3<br>OutValue = 12 |

### Ragne

This module outputs the specified high value when the input value is greater than the specified high value into the output node, or outputs the specified low value when the input value is less than the specified 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 | if (ExecutionNode is not connected) {<br>   if (InValue < User specified low value)<br>      OutValue = User specified low value<br>   else<br>      OutValue = InValue<br>   if (User specified high value < InValue)<br>      OutValue = User specified high value<br>   else<br>      OutValue = InValue<br>}<br>else {<br>   if (ExecutionNode data is “true”) {<br>      if (InValue M ¥¥< User specified low data)<br>         OutValue = User specified log value<br>      else<br>         OutValue = InValue<br>      if (User specified high value < InValue)<br>         OutValue = User specified high value<br>      else<br>         OutValue = InValue<br>   }<br>   else<br>      OutValue = InValue<br>} |
| Example | InValue = 12.3<br>OutValue = 12 |

## 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<br>InValue2 |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | if (InValue1 == InValue2)<br>    OutValue = “true”<br>else<br>    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<br>InValue2 |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | if (InValue1 != InValue2)<br>    OutValue = “true”<br>else<br>    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<br>InValue2 |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | if (InValue1 > InValue2)<br>   OutValue = “true”<br>else<br>   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<br>InValue2 |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | if (InValue1 < InValue2)<br>   OutValue = “true”<br>else<br>   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<br>InValue2 |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | if (InValue1 >= InValue2)<br>   OutValue = “true”<br>else<br>   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<br>InValue2 |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | if (InValue1 <= InValue2)<br>   OutValue = “true”<br>else<br>   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 | if (ExecutionNode is not connected)<br>    OutValue = ! InValue<br>else {<br>    if (ExecutionNode data is “true”) {<br>        OutValue = ! InValue<br>    else<br>        OutValue = InValue<br>} |

## 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 | InValue1<br>InValue2 |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | O7utValue = InValue1 && InValue2 |

## 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 | InValue1<br>InValue2 |
| Output node names | OutValue |

| | |
|---|---|
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | OutValue = InValue1 \|\| InValue2 |

## *Geometry*

### Normalize

This module outputs a normalized vector of the input node 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 outputs a negated vector of the input node 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 outputs a length of the input vector 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 outputs a inner product of two input vectors into the output node.

| | |
|---|---|
| Input node names | InValue (x1, y1, z1)<br>InValue (x2, y2, z2) |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | OutValue = (x1*x2) + (y1*y2) + (z1*z2) |

## GetAngle

This module outputs a radian angle between two input vectors into the output node.

| | |
|---|---|
| Input node names | InValue (x1, y1, z1)<br>InValue (x2, y2, z2) |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | OutValue = acos( ((x1*x2) + (y1*y2) + (z1*z2)) /<br>(sqrt(x1*x1+y1*y1+z1*z1) + (x2*x2+y2*y2+z2*z2) ) |

## GetVector

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

| | |
|---|---|
| Input node names | InValue (x1, y1, z1)<br>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 outputs a distance between two input vectors into the output node.

| | |
|---|---|
| Input node names | InValue (x1, y1, z1)<br>InValue (x2, y2, z2) |
| Output node names | OutValue |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | OutValue = sqrt((x2-x1)^2 + (y2-y2)^2 + (x2-x1)^2) |

## GetCross

This module outputs a cross vector of two input vectors into the output node.

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

## Rotate

This module rotates a vector of the first input node by a rotation of the second input node, then outputs the rotated vector into the output node.

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

## *Transform*

Modules of this class set or get a field value of the specified Transform node. Double-click the module or input a string of the node name into the first input node to specify the Transform node, The specified node name is drawn under the module.

When the module has an execution node and the execution node receives a "false" string, the module doesn't execute the operation.

### GetName

This module outputs a name of the specified Transform node into the output node.

| | |
|---|---|
| Input node names | |
| Output node names | Node name |
| Execution node | - |
| Setting dialog | O |
| Target node | Transform |

### SetTranslation

This module sets a value of the second input node into a translation field of the specified Transform node.

| | |
|---|---|
| Input node names | Node name<br>Translation (x, y, z) |
| Output node names | -, |
| Execution node | O |
| Setting dialog | O |
| Target node | Transform |
| Result | Transform : : translation = translation |

### SetRotation

This module sets a value of the second input node into a rotation field of the specified Transform node.

| | |
|---|---|
| Input node names | Node name<br>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 a value of the second input node into a scale field of the specified Transform node.

| | |
|---|---|
| Input node names | Node name<br>Scale (x, y, z) |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Transform |
| Result | Transform : : scale= Scale |

## SetCenter

This module sets a value of the second input node into a center field of the specified Transform node.

| | |
|---|---|
| Input node names | Node name<br>Center (x, y, z) |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Transform |
| Result | Transform : : center = center |

## GetTranslation

This module outputs a translation field value of the specified Transform node into the output node.

| | |
|---|---|
| Input node names | Node name |
| Output node names | Translation (x, y, z) |
| Execution node | - |

| | |
|---|---|
| Setting dialog | O |
| Target node | Transform |
| Result | Translation = Transform : : translation |

## GetRotation

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

| | |
|---|---|
| Input node names | Node name |
| 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 scale field value of the specified Transform node into the output node.

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

## GetCenter

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

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

## Material

Modules of this class set or get a field value of the specified Material node. Double-click the module or input a string of the node name into the first input node to specify the Material node. The specified node name is drawn under the module.

When the module has an execution node and the execution node receives a “false” string, the module doesn’t execute the operation.

### GetName

This module outputs a name of the specified Material node into the output node.

| | |
|---|---|
| Input node names | |
| Output node names | Node name |
| Execution node | - |
| Setting dialog | O |
| Target node | Material |

### SetAmbientIntensity

This module sets a value of the second input node into a ambientIntensity field of the specified Material node.

| | |
|---|---|
| Input node names | Node name<br>AmbientIntensity |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Material |
| Result | Material : : ambientIntensity = AmbientIntensity |

### SetDiffuseColor

This module sets a value of the second input node into a diffuseColor field of the specified Material node.

| | |
|---|---|
| Input node names | Node name<br>DiffuseColor (r, g, b) |

| | |
|---|---|
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Material |
| Result | Material : : diffuseColor = DiffuseColor |

## SetEmissiveColor

This module sets a value of the second input node into a emissiveColor field of the specified Material node.

| | |
|---|---|
| Input node names | Node name<br>EmissiveColor (r, g, b) |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Material |
| Result | Material : : emissiveColor = EmissiveColor |

## SetSpecularColor

This module sets a value of the second input node into a specularColor field of the specified Material node.

| | |
|---|---|
| Input node names | Node name<br>SpecularColor (r, g, b) |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Material |
| Result | Material : : specularColor = SpecularColor |

## SetShininess

This module sets a value of the second input node into a shininess field of the specified Material node.

| | |
|---|---|
| Input node names | Node name<br>Shininess |
| Output node names | - |

| | |
|---|---|
| Execution node | O |
| Setting dialog | O |
| Target node | Material |
| Result | Material : : shininess = Shininess |

## SetTransparency

This module sets a value of the second input node into a transparency field of the specified Material node.

| | |
|---|---|
| Input node names | Node name<br>Transparency |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Material |
| Result | Material : : transparency = Transparency |

## GetAmbientIntensity

This module outputs a ambientIntensity field value of the specified Material node into the output node.

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

## GetDiffuseColor

This module outputs a diffuseColor field value of the specified Material node into the output node.

| | |
|---|---|
| Input node names | Node name |
| Output node names | DiffuseColor (r, g, b) |
| Execution node | - |
| Setting dialog | O |

| | |
|---|---|
| Target node | Material |
| Result | DiffuseColor = Material : : diffuseColor |

## GetEmissiveColor

This module outputs a emissiveColor field value of the specified Material node into the output node.

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

## GetSpecularColor

This module outputs a specularColor field value of the specified Material node into the output node.

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

## GetShininess

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

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

## GetTransparency

This module outputs a transparency field value of the specified Material node into the output node.

| | |
|---|---|
| Input node names | Node name |
| Output node names | Transparency |
| Execution node | - |
| Setting dialog | O |
| Target node | Material |
| Result | Transparency = Material : : transparency |

## Light

Modules of this class set or get a field value of the specified DirectionalLight, PointLight or SpotLight node. Double-click the module or input a string of the node name into the first input node to specify the light node, The specified node name is drawn under the module.

When the module has an execution node and the execution node receives a “false” string, the module doesn’t execute the operation.

### GetName

This module outputs a name of the specified DirectionalLight, PointLight or SpotLight node into the output node.

| | |
|---|---|
| Input node names | |
| Output node names | Node name |
| Execution node | - |
| Setting dialog | O |
| Target node | DirectionalLight / PointLight / SpotLight |

### SetOn

This module sets a value of the second input node into an on field of the specified DirectionalLight, PointLight or SpotLight node.

| | |
|---|---|
| Input node names | Node name<br>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 a value of the second input node into a color field of the specified DirectionalLight, PointLight or SpotLight node.

| | |
|---|---|
| Input node names | Node name<br>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 a value of the second input node into a intensity field of the specified DirectionalLight, PointLight or SpotLight node.

| | |
|---|---|
| Input node names | Node name<br>Intensity |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | DirectionalLight / PointLight / SpotLight |
| Result | Light : : intensity = Intensity |

## SetLocation

This module sets a value of the second input node into a on location of the specified PointLight or SpotLight node.

| | |
|---|---|
| Input node names | Node name<br>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 a value of the second input node into a direction field of the specified DirectionalLight or SpotLight node.

| | |
|---|---|
| Input node names | Node name<br>Node name 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 a value of the second input node into a on field of the specified PointLight or SpotLight node.

| | |
|---|---|
| Input node names | Node name<br>Radius |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | PointLight / SpotLight |
| Result | Light : : radius = Radius |

## GetOn

This module outputs an on field value of the specified DirecionalLight, PointLight or SpotLight node into the output node.

| | |
|---|---|
| Input node names | Node name |
| 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 color field value of the specified DirecionalLight, PointLight or SpotLight node into the output node.

| | |
|---|---|
| Input node names | Node name |
| 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 an intensity field value of the specified DirecionalLight, PointLight or SpotLight node into the output node.

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

## GetLocation

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

| | |
|---|---|
| Input node names | Node name |
| 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 direction field value of the specified DirecionalLight or SpotLight node into the output node.

| | |
|---|---|
| Input node names | Node name |
| 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 radius field value of the specifiedPointLight, or SpotLight node into the output node.

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

## Viewpoint

Modules of this class set or get a field value of the specified Viewpoint node. Double-click the module or input a string of the node name into the first input node to specify the Viewpoint node, The specified node name is drawn under the module.

When the module has an execution node and the execution node receives a “false” string, the module doesn’t execute the operation.

### GetName

This module outputs a name of the specified Viewpoint node into the output node.

| | |
|---|---|
| Input node names | |
| Output node names | Node name |
| Execution node | - |
| Setting dialog | O |
| Target node | Viewpoint |

### SetPosition

This module sets a value of the second input node into a position field of the specified Viewpoint node.

| | |
|---|---|
| Input node names | Node name<br>Position (x, y, z) |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Viewpoint |
| Result | Viewpoint : : position = Position |

### SetOrientation

This module sets a value of the second input node into a orientation field of the specified Viewpoint node.

| | |
|---|---|
| Input node names | Node name<br>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 a value of the second input node into a fieldOfView field of the specified Viewpoint node.

| | |
|---|---|
| Input node names | Node name<br>fov |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Viewpoint |
| Result | Viewpoint : : fieldOfView = fov |

## GetPosition

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

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

## GetOrientation

This module outputs an orientation field value of the specified Viewpoint node into the output node.

| | |
|---|---|
| Input node names | Node name |
| 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 fieldOfView field value of the specified Viewpoint node into the output node.

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

## ***Interpolator***

Modules of this class play or stop an interpolator node as an animation sequence like a VCR. Double-click the module or input a string of the node name into the first input node to specify the Interpolator node, The specified node name is drawn under the module.

When a set_fraction field of the Interpolator node is changed using the modules, the Interpolator node sets a new value into the value_changed field, then ROUTEs that are connected with the value_changed field is updated automatically.

### GetName

This module outputs a name of the specified Interpolator node into the output node.

| | |
|---|---|
| Input node names | |
| Output node names | Node name |
| Execution node | - |
| Setting dialog | O |
| Target node | CoordinateInterpolator / NormalInterpolator / OrientationInterpolator / PositionInterpolator / ScalarInterpolator |

### Play

This module begins the playback of the specified Interpolator node from the current fraction position when the execution node receives a “true” string. You can set the play mode using the setting dialog.

| | |
|---|---|
| Input node names | Node name |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | CoordinateInterpolator / NormalInterpolator / OrientationInterpolator / PositionInterpolator / ScalarInterpolator |

## Stop

This module stops the playback of the specified Interpolator node when the execution node receives a “true” string, and sets the current fraction position to 0.

| | |
|---|---|
| Input node names | Node name |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | CoordinateInterpolator / NormalInterpolator / OrientationInterpolator / PositionInterpolator / ScalarInterpolator |

## Pose

This module stops the playback of the specified Interpolator node when the execution node receives a “true” string.

| | |
|---|---|
| Input node names | Node name |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | CoordinateInterpolator / NormalInterpolator / OrientationInterpolator / PositionInterpolator / ScalarInterpolator |

## IsPlaying

This module outputs a “true” string when the specified Interpolator node is playing now, otherwise it outputs a “false” string.

| | |
|---|---|
| Input node names | Node name |
| Output node names | IsPlaying |
| Execution node | O |
| Setting dialog | O |
| Target node | CoordinateInterpolator / NormalInterpolator / OrientationInterpolator / PositionInterpolator / ScalarInterpolator |

## Rewind

This module sets the current fraction position to 0 when the execution node receives a “true” string.

| | |
|---|---|
| Input node names | Node name |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | CoordinateInterpolator / NormalInterpolator /<br>OrientationInterpolator / PositionInterpolator /<br>ScalarInterpolator |

## Next

This module adds a second input value to the current fraction position.

| | |
|---|---|
| Input node names | Node name<br>Step |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | CoordinateInterpolator / NormalInterpolator /<br>OrientationInterpolator / PositionInterpolator /<br>ScalarInterpolator |
| Result | Interpolator :: fraction += Step |

## Prev

This module subtracts a second input value to the current fraction position.

| | |
|---|---|
| Input node names | Node name<br>Step |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | CoordinateInterpolator / NormalInterpolator /<br>OrientationInterpolator / PositionInterpolator /<br>ScalarInterpolator |

| | |
|---|---|
| Result | Interpolator :: fraction -= Step |

## SetFraction

This module sets a second input value to the current fraction position.

| | |
|---|---|
| Input node names | Node name<br>Fraction |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | CoordinateInterpolator / NormalInterpolator / OrientationInterpolator / PositionInterpolator / ScalarInterpolator |
| Result | Interpolator :: fraction = Fraction |

## GetFraction

This module outputs the current fraction position into the output node..

| | |
|---|---|
| Input node names | Node name |
| Output node names | Fraction |
| Execution node | O |
| Setting dialog | O |
| Target node | CoordinateInterpolator / NormalInterpolator / OrientationInterpolator / PositionInterpolator / ScalarInterpolator |
| Result | Fraction = Interpolator :: fraction |

## Sensor

Modules of this class output a record of virtual reality devices. Currently, CyberToolbox supports as the following devices

Logitech Magellan (Spacecontrol Mouse)
Polhemus Fastrak / Isotrak2
Intersense IS300
BG Systems BeeBox
Joystick (only WIN32 platform)
Mouse

If you want to use the device that is connected with RS-232C ports, you have to install Sun's Java Communications API.

### Mouse

This module outputs current mouse records into the output nodes.

| | |
|---|---|
| Input node names | |
| Output node names | Button<br>Position (x, y) |
| Execution node | - |
| Setting dialog | - |

### Magellan

This module outputs current records of the specified Magellan into the output nodes.

| | |
|---|---|
| Input node names | |
| Output node names | Button<br>Translation (x, y, z)<br>Radian Angle (x, y, z) |
| Execution node | - |
| Setting dialog | O |

## Fastrak

This module outputs current records of the specified receiver of Fastrak into the output nodes.

| | |
|---|---|
| Input node names | |
| Output node names | Position (x, y, z)<br>Radian Angle (x, y, z) |
| Execution node | - |
| Setting dialog | O |

## Isotrak2

This module outputs current records of the specified receiver of Isotrak2 into the output nodes.

| | |
|---|---|
| Input node names | |
| Output node names | Position (x, y, z)<br>Radian Angle (x, y, z) |
| Execution node | - |
| Setting dialog | O |

## IS300

This module outputs current records of the specified receiver of IS300 into the output nodes.

| | |
|---|---|
| Input node names | |
| Output node names | Position (x, y, z)<br>Radian Angle (x, y, z) |
| Execution node | - |
| Setting dialog | O |

## Joystick

This module outputs current records of the specified joystick into the output nodes.

| | |
|---|---|
| Input node names | |
| Output node names | Button<br>Translation (x, y) |
| Execution node | - |
| Setting dialog | O |

## BeeBox

This module outputs current records of the specified BeeBox into the output nodes.

| | |
|---|---|
| Input node names | |
| Output node names | Button<br>Stick (x, y)<br>Lever |
| Execution node | - |
| Setting dialog | O |

## *Misc*

### SetMessage

This module sends a second input value into the specified event. Using the module, yon can send a message to your original events.

| | |
|---|---|
| Input node names | Event name<br>Message |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |

### InsideDiagram

This module can have a diagram inside the module. To edit the inside diagram, double-click the module.

| | |
|---|---|
| Input node names | InNode1<br>InNode2<br>InNode3<br>InNode4 |
| Output node names | OutNode1<br>OutNode2<br>OutNode3<br>OutNode4 |
| Execution node | O |
| Setting dialog | - |

### SetData

This module sets a value of the second input node into a data of the specified data name.

| | |
|---|---|
| Input node names | Data name<br>value |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | - |

| | |
|---|---|
| Result | Data name : : value = value |

## GetData

This module outputs a value of the specified data name.

| | |
|---|---|
| Input node names | Data name |
| Output node names | value- |
| Execution node | O |
| Setting dialog | O |
| Target node | - |
| Result | value = Data name : : value |

## SetArrayData

This module sets a value of the second input node into a data of the specified group name and data name.

| | |
|---|---|
| Input node names | Group name<br>Data name<br>value |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | - |
| Result | (Group name, Data name) : : value = value |

## GetArrayData

This module outputs a value of the specified group name and data name.

| | |
|---|---|
| Input node names | Group name<br>Data name |
| Output node names | value- |
| Execution node | O |
| Setting dialog | O |
| Target node | - |
| Result | value = (Group name, Data name) : : value |

## SetSwitch

This module sets a value of the input node into a whichChoice field of the specified Switch node.

| | |
|---|---|
| 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 a value of the input node into a skyColor field of the specified Background node.

| | |
|---|---|
| Input node names | color |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Background |
| Result | Background : : skyColor = color |

## SetText

This module sets a value of the input node into a string field of the specified Text node.

| | |
|---|---|
| Input node names | text |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | Text |
| Result | Text : : string[0] = text |

## GetTime

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

| | |
|---|---|
| Input node names | - |
| Output node names | Hour<br>Minute<br>Second |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | Hour = current system hour<br>Minute = current system minute<br>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 |

## PlaySound

This module play the specified sound when the execution node is received a “true” string.

| | |
|---|---|
| Input node names | - |
| Output node names | - |
| Execution node | - |
| Setting dialog | O |
| Target node | - |
| Result | Play the specified sound |

## 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 |

## ShowDocument

This module requests that the browser or applet viewer show a specified Web page.

| | |
|---|---|
| Input node names | URL<br>Target |
| Output node names | - |
| Execution node | O |
| Setting dialog | O |
| Target node | - |

## ShowStatus

This module requests that a specified string be displayed in the "status window".

| | |
|---|---|
| Input node names | String |
| Output node names | - |
| Execution node | O |
| Setting dialog | - |
| Target node | - |

## 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 | - |

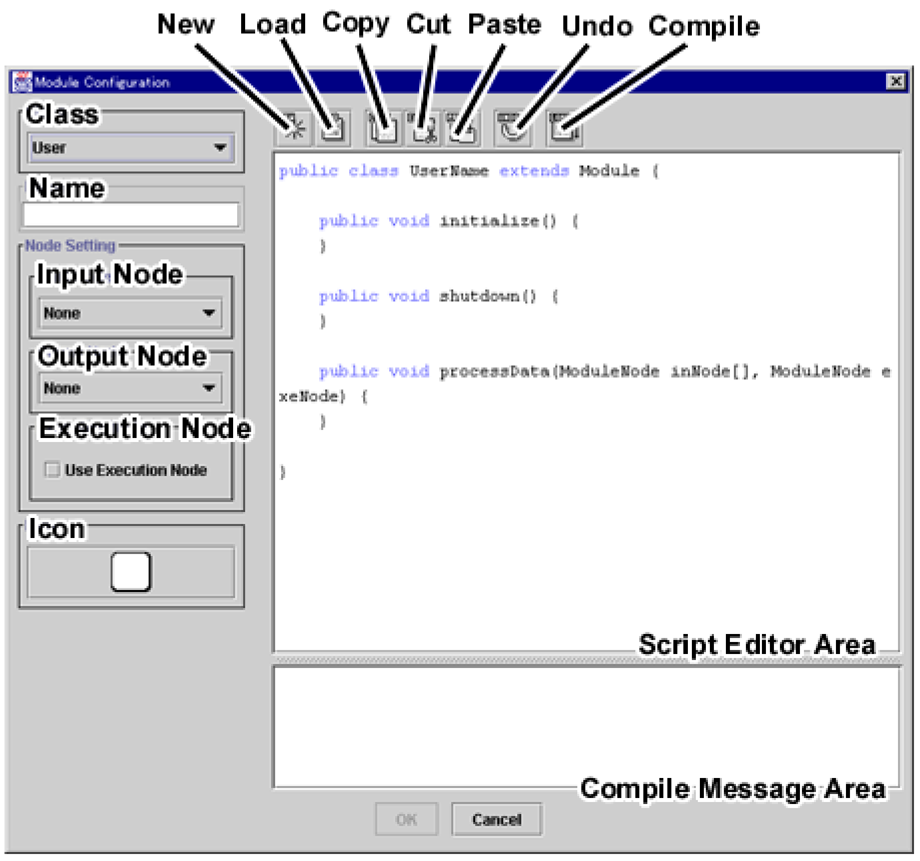
## GetScreenSize

This module outputs a size of a perspective window into the output node.

| | |
|---|---|
| Input node names | - |
| Output node names | ScreenSize |
| Execution node | - |
| Setting dialog | - |
| Target node | - |
| Result | ScreenSize = width,height |

## Adding your original modules

Using “New Module” button in the world window, you can add your original modules into CyberToolbox. To click the button, the following dialog is opened.



Class: User

Input Node: None

Output Node: None

☐ Use Execution Node

```
public class UserName extends Module {

    public void initialize() {
    }

    public void shutdown() {
    }

    public void processData(ModuleNode inNode[], ModuleNode e
xeNode) {
    }

}
```

To add the new module, you have to set the following dialog fields, write a behavior code of the module, and compile the code normally.

| Dialog Field | Description |
|---|---|
| Class | Select a class type of the module. |
| Name | Set a name of the module. |
| Input Node | Select a number of the input nodes. |
| Output Node | Select a number of the output nodes. |
| Execution Node | Check the box whether the module has an execution node. |
| Icon | Select an icon image of the module. The image size has to 32z32, and the image format is GIF. |

## ***Module Class***

All modules of CyberToolbox are sub class of Module class. Therefor, a code of adding module has to be a sub class of the Module class too. Because the Module class is an abstract class, you have to define the following three methods.

### ***public void initialize()***

This method is called before the simulation is started. Use this method to write initialization codes.

### ***public void shutdown()***

This method is called after the simulation is stopped. Use this method to write termination codes.

### ***public void processData(ModuleNode inNode[], ModuleNode exeNode)***

This method is called when the module is received data from other modules, or the module is activated.

The inNode argument has the data of the input nodes, and the array length is same as the number of the input nodes of the module. If the module doesn't have the input nodes, the argument is null.

The exeNode argument has the data of the execution node. If the module doesn't have the execution node, the argument is null.

Using the following methods of the ModuleNode class, you can get the node information.

```
public class ModuleNode {
    public boolean isConnected()
    public String getStringValue()
    public int getIntegerValue() throws NumberFormatException
    public float getFloatValue() throws NumberFormatException
    public double getDoubleValue() throws NumberFormatException
    public boolean getBooleanValue()
    public String []getStringValues()
    public double []getDoubleValues()
    public float []getFloatValues()
    public int []getIntegerValues()
}
```

Use the isConnected() to know whether the node is connected with a output node of the other module by a data-flow line. If the node is not connected, you should ignore data of the node.

Using data of the input and execution nodes, you have to send new data to other modules through the output nodes. Use the following methods of the Module class to send the data into the output nodes.

```
public void sendOutNodeValue(int n, String value)
public void sendOutNodeValue(int n, int value)
public void sendOutNodeValue(int n, float value)
public void sendOutNodeValue(int n, double value)
public void sendOutNodeValue(int n, boolean value)
public void sendOutNodeValue(int n, String value[])
public void sendOutNodeValue(int n, int value[])
public void sendOutNodeValue(int n, float value[])
public void sendOutNodeValue(int n, double value[])
```

## ***Example 1   Misc::GetTime***

This module has no the input nodes and the execution node, three output nodes to output current time.

```
public class MiscGetTime extends Module {
    Calendar calendar = new GregorianCalendar();
    public void initialize() {
    }
    public void shutdown() {
    }
    public void processData(ModuleNode inNode[], ModuleNode exeNode){
           calendar.setTime(new Date());
           sendOutNodeValue(0, calendar.get(Calendar.HOUR_OF_DAY));
           sendOutNodeValue(1, calendar.get(Calendar.MINUTE));
           sendOutNodeValue(2, calendar.get(Calendar.SECOND));
    }
}
```

## ***Example 2 Math::Abs***

This module has an input node, a output node, and a execution node. The output node outputs an absolute value of the input node when the execution node is not connected or the execution node received a true value. Otherwise the module outputs data of the input node as it is.

```
public class MathAbs extends Module {
    public void initialize() {
    }
    public void shutdown() {
    }
    public void processData(ModuleNode inNode[], ModuleNode exeNode){
           if (exeNode.isConnected() == true) {
                  if (exeNode.getBooleanValue() == false) {
                         sendOutNodeValue(0,
                                inNode[0].getStringValue());
                         return;
                  }
           }
           try {
                  double value = inNode[0].getDoubleValue();
                  sendOutNodeValue(0, Math.abs(value));
           }
           catch (NumberFormatException nfe) {
                  sendOutNodeValue(0, Double.toString(Double.NaN));
           }
    }
}
```

# Supported File Formats

CyberToolbox supports for loading the following geometry file formats.

## VRML97

CyberToolbox can gets all information in the specified VRML97 file, add the nodes into the scene graph.

## Autodesk 3DS

CyberToolbox gets only the following information from the specified 3DS file, and ignore the other information. CyberToolbox converts from the loading information into VRML97 nodes, add the nodes into the scene graph.

| Chunk ID | Description |
|---|---|
| 0xA010 | Material Ambient Color |
| 0xA020 | Material Diffuse Color |
| 0xA030 | Material Specular Color |
| 0xA040 | Materisl Shininess |
| 0x4100 | Triangle Set |
| 0x4110 | Triangle Point Set |
| 0x4120 | Triangle Fase Set |

## Wavefront OBJ

CyberToolbox gets only the following information from the specified OBJ file, and ignore the other information. CyberToolbox doesn’t read the map files and the material files. CyberToolbox converts from the loading information into VRML97 nodes, add the nodes into the scene graph.

| ID | Description |
|---|---|
| v | Vertex Position |
| vn | Vertex Normal |
| f | Face Index |

## ***LightWave3D LWS***

CyberToolbox uses a utility class of Java3D package, com.sun.j3d.loaders.lw3d , for loading LWS files. Please see the document of the package about the loader in more detail. CyberToolbox converts from Java3D nodes that are loaded by the package into the VRML97 nodes, add the nodes into the scene graph.

## ***SENSE8 NFF***

CyberToolbox gets only the vertex positions and the polygon indices with the color from the specified NFF file, and ignore the other information. CyberToolbox converts from the loading information into VRML97 nodes, add the nodes into the scene graph.

# Applet of CyberToolbox

Users of CyberToolbox can distribute their contents that are created by CyberToolbox on the Internet. For the purpose, an applet package of CyberToolbox is distributed free. The package file name is “ctbj3d.jar”, you can download the latest package from same site of CyberToolbox.

## Setting for Appletviewer

To browse the contents using appletviewer utility that is included in JDK1.2, you have to use APPLET tag, and set “WorldApplet.class” into the CODE parameter, “ctbj3d.jar” into the ARCHIVE parameter, and specify a file or URL name of your contents using the PARAM tag as the following.

```
<HTML>
<BODY>
<CENTER>
<H1>CyberToolbox Release 3.0</H1>
<APPLET CODE="WorldApplet.class" ARCHIVE="ctbj3d.jar"
   width=250 height=250>
   <PARAM NAME=src VALUE="world/sthenge/sthenge.wrl">
</APPLET>
</CENTER>
</BODY>
</HTML>
```

About the PARAM tag, you have to set only a file or URL name for a VRML97 file of the contents to load into the applet. You don’t need set the behavior file of the contents too because the information is included in the VRML97 file using the WorldInfo node. For example,

```
DEF CTB_BEHAVIRO_INFO WorldInfo {
    title "CyberToolbox Behavior Format V2.0"
    info [
           "sthenge.cbf"
    ]
}
```

## Setting for Microsoft or Netscape Browsers

If you want to browse the contents using Microsoft Internet Explore or Netscape Communicator, you have to install Sun’s Java Plugin.

Then you have to point Java Plugin to use the JDK1.2 or JRE1.2, and add the following four class files into your CLASSPATH.

j3dcore.jar, j3dutils.jar, vecmath.jar, j3daudio.jar

To browse the contents using the browsers, you have to use OBJECT tag instead of APPLET tag for Java Plug-In. For example,

```
<HTML>
<BODY>
<CENTER>
<H1>CyberToolbox for Java3D</H1>
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
   WIDTH=250 HEIGHT=250
   CODEBASE="http://java.sun.com/products/plugin/1.2/
                   jinstall-12-win32.cab#Version=1,2,0,0">
   <PARAM NAME=CODE VALUE="WorldApplet.class" >
   <PARAM NAME=ARCHIVE VALUE="ctbj3d.jar" >
   <PARAM NAME="type" VALUE="application/x-java-applet;version=1.2">
   <PARAM NAME=src VALUE="world/sthenge/sthenge.wrl">
</OBJECT>
</CENTER>
</BODY>
</HTML>
```

If you want to set the OBJECT tag easily, you should convert from HTML files for appletviewer using Sun’s HTML Converter for Java Plugin.